Designing Local Algorithms with Algorithms

Jukka Suomela · Aalto University
Joint work with...

Juho Hirvonen · Janne H. Korhonen
Tuomo Lempiäinen · Christopher Purcell
Joel Rybicki · Patric Östergård (Aalto)

Sebastian Brandt · Przemysław Uznański (ETH Zurich)

Orr Fischer (Tel Aviv)
Algorithm synthesis

• Computer science: what can be automated?
• Can we *automate our own work*?
• Can we outsource algorithm design to computers?
  • **input**: problem specification
  • **output**: asymptotically optimal algorithm
Today: a success story

• Case study:
  • computational design of local distributed algorithms for LCL problems on grid graphs

• Spoiler:
  • undecidable – but with one bit of advice we can do it!
  • not just in theory but also in practice
Setting

- Distributed graph algorithms
- **Input graph = computer network**
  - node = computer, edge = communication link
  - unknown topology
- Each node outputs its own part of solution
  - e.g. graph colouring: node outputs its own colour
Setting

• Deterministic distributed algorithms, LOCAL model of computing
  • unique identifiers
  • synchronous communication rounds
  • time = number of rounds until all nodes stop
  • unlimited message size, unlimited local computation
Setting

- Deterministic distributed algorithms, **LOCAL** model of computing
- Time = distance
- Algorithm with running time $T$: *mapping from radius-$T$ neighbourhoods to local outputs*
LCL problems

• LCL = locally checkable labelling
  • Naor–Stockmeyer (1995)

• Valid solution can be detected by checking $O(1)$-radius neighbourhood of each node
  • maximal independent set, maximal matching, vertex colouring, edge colouring …
LCL problems

• All LCL problems can be solved with $O(1)$-round \textit{nondeterministic} algorithms
  • guess a solution, verify it in $O(1)$ rounds

• Key question: how fast can we solve them with \textit{deterministic} algorithms?
  • cf. P vs. NP
Traditional settings

• **Directed cycles**
  • Cole–Vishkin (1986), Linial (1992)…
  • well understood

• **General (bounded-degree) graphs**
  • lots of ongoing work…
  • typical challenge: *expander-like constructions*
Our setting today

- **Oriented grids** (2D)
  - toroidal grid, $n \times n$ nodes, unique identifiers
  - consistent orientations north/east/south/west

- **Generalisation of directed cycles** (1D)

- Closer to real-world systems than expander-like worst-case constructions?
Warm-up examples

• Vertex colouring in 1D grids

• **2-colouring**: global, $\Theta(n)$ rounds

• **3-colouring**: local, $\Theta(\log^* n)$ rounds
  • Cole–Vishkin (1986), Linial (1992)
Why is 3-colouring $\Theta(\log^* n)$?

• Upper bound: one-round colour reduction
  • input: colouring with $2^k$ colours
  • output: colouring with $2k$ colours

• Lower bound: speed-up lemma
  • given: algorithm for $k$-colouring in time $T$
  • construct: algorithm for $2^k$-colouring in time $T - 1$
Warm-up examples

- Vertex colouring in 1D grids
- 2-colouring: global, $\Theta(n)$ rounds
- 3-colouring: local, $\Theta(\log^* n)$ rounds
Warm-up examples

- Vertex colouring in 2D grids
- 2-colouring: global, $\Theta(n)$ rounds
- 3-colouring: ???
- 4-colouring: ???
- 5-colouring: local, $\Theta(\log^* n)$ rounds
Warm-up examples

- Vertex colouring in 2D grids

- **2-colouring:** global, $\Theta(n)$ rounds

- **3-colouring:** global, $\Theta(n)$ rounds

- **4-colouring:** local, $\Theta(\log^* n)$ rounds

- **5-colouring:** local, $\Theta(\log^* n)$ rounds
Warm-up examples

- Vertex colouring in 4-regular graphs
- **2-colouring**: global, $\Theta(n)$ rounds
- **3-colouring**: global, $\Theta(n)$ rounds
- **4-colouring**: intermediate, polylog rounds
- **5-colouring**: local, $\Theta(\log^* n)$ rounds
Complexity of LCL problems

• 1D grids:
  • everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  • decidable

• Bounded-degree graphs:
  • intermediate complexities, $\text{polylog}(n)$ … (Brand et al. 2016)
  • undecidable (Naor–Stockmeyer 1995)
Complexity of LCL problems

• 1D grids:
  • everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  • decidable

• 2D grids:
  • everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  • undecidable
Complexity of LCL problems

• 1D grids:
  • everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  • decidable

• 2D grids:
  • everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  • undecidable — but let us not despair!
Goal: algorithm synthesis

• Setting:
  • input: specification of an LCL problem
  • output: asymptotically optimal algorithm for 2D grids

• Does this make any sense?
  • most interesting case: $\Theta(\log^* n)$ time
  • how could one even represent an arbitrary $\Theta(\log^* n)$-round algorithm in a computer??
$O(\log^* n)$
Goal: algorithm synthesis

• $\Theta(\log^* n)$-round algorithm in 2D grids:
  • mapping from $\Theta(\log^* n) \times \Theta(\log^* n)$ neighbourhoods to local outputs
  • nodes are labelled with 1, 2, …, poly$(n)$

• Infinite family of functions

• Awkward to handle with computers
Key insight: normalisation

- **Setting**: LCL problems, 2D grids

- **Theorem**: Any $o(n)$-time algorithm can be translated to a “normal form”
  - we isolate a fixed $\Theta(\log^* n)$-time component
  - everything else is a finite function
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>33</td>
<td>77</td>
<td>57</td>
<td>49</td>
<td>26</td>
</tr>
<tr>
<td>71</td>
<td>79</td>
<td>8</td>
<td>62</td>
<td>48</td>
<td>24</td>
</tr>
<tr>
<td>31</td>
<td>21</td>
<td>15</td>
<td>30</td>
<td>60</td>
<td>67</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>17</td>
<td>95</td>
<td>23</td>
<td>47</td>
</tr>
<tr>
<td>87</td>
<td>80</td>
<td>25</td>
<td>38</td>
<td>20</td>
<td>64</td>
</tr>
<tr>
<td>45</td>
<td>61</td>
<td>91</td>
<td>51</td>
<td>69</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>53</td>
<td>63</td>
<td>40</td>
<td>16</td>
<td>2</td>
</tr>
</tbody>
</table>

$O(\log^* n)$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$O(1)$
Key insight: normalisation

• For any problem $P$ of complexity $\Theta(\log^* n)$, there are constants $k$ and $r$ and function $f$ such that $P$ can be solved as follows:
  • input: 2D grid $G$ with unique identifiers
  • find a maximal independent set in $G^k$
  • discard unique identifiers
  • apply function $f$ to each $r \times r$ neighbourhood
Some proof ideas

• Given: \(A\) solves \(P\) in time \(o(n)\) in \(n \times n\) grids

• Solving \(P\) in time \(O(\log^* N)\) in \(N \times N\) grids:
  • pick suitable \(n = O(1), k = O(1)\)
  • find MIS in \(G^k\)
  • use MIS to find \textit{locally unique identifiers} for \(n \times n\) neighbourhoods
  • simulate \(A\) in \(n \times n\) local neighbourhoods
Normalisation in practice

• Example: 4-colouring

• Sufficient to pick $k = 3$, $r = 7$

• Algorithm $\approx$ mapping $\{0, 1\}^{7 \times 7} \rightarrow \{1, 2, 3, 4\}$
  • only finitely many candidates
  • given a candidate, we can easily verify if it is good
What about undecidability?

- **Trivial case:** complexity $O(1)$
- **Undecidable:** given an LCL problem, is its complexity $\Theta(\log^* n)$ or $\Theta(n)$ in 2D grids?
- However, if we get just *one bit of advice* (or make a lucky guess), we can find an asymptotically optimal algorithm!
Synthesis with advice

- **Advice**: complexity is $\Theta(\log^* n)$
  - try each pair $(r, k)$
  - check if there is a valid mapping from binary $r \times r$ matrices that represent local parts of maximal independent sets in $G^k$

- **Advice**: complexity is $\Theta(n)$
  - trivial brute force is optimal
It works in practice, too!

- **Ongoing work:** we have already synthesised asymptotically optimal algorithms for *thousands* of LCL problems
  - “high-throughput algorithm design”
  - can gain insights into the structure of large families of *parametrised problems*
  - synthesis unsuccessful: conjecture lower bound?
Some building blocks

• Enumerate all $r \times r$ neighbourhoods that represent possible fragments of maximal independent sets in $G^k$

• Construct neighbourhood graphs
  • algorithm $\approx$ labelling of neighbourhood graph

• Apply SAT solvers to find a labelling
Human beings still needed

• Computers can design e.g. very efficient algorithms for 4-colouring

• We still needed human beings to prove that there is no algorithm for 3-colouring
  • new lower-bound techniques needed
3-colouring is hard: proof idea

- W.l.o.g., consider greedy 3-colouring
  - 2-coloured regions + oriented boundaries

- Boundaries form closed curves
  - sum of *signed boundary crossings* is preserved

- We could solve a global problem on cycles:
  - “constant-sum \{-1, 0, +1\} labelling”
Conclusions

• Nontrivial algorithms: $\Theta(\log^* n)$ complexity

• Any such algorithm can be split in two parts:
  • “symmetry breaking”: find an MIS
  • “computation”: nontrivial but finite

• Main open question: how far can we push this beyond oriented 2D grids?
<table>
<thead>
<tr>
<th></th>
<th>92</th>
<th>33</th>
<th>77</th>
<th>57</th>
<th>49</th>
<th>26</th>
<th>74</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>79</td>
<td>8</td>
<td>62</td>
<td>48</td>
<td>24</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>21</td>
<td>15</td>
<td>30</td>
<td>60</td>
<td>67</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>17</td>
<td>95</td>
<td>23</td>
<td>47</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>80</td>
<td>25</td>
<td>38</td>
<td>20</td>
<td>64</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>61</td>
<td>91</td>
<td>51</td>
<td>69</td>
<td>1</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>53</td>
<td>63</td>
<td>40</td>
<td>16</td>
<td>2</td>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>

$O(\log^* n)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$O(1)$

$\text{MIS}$

$\text{MIS}$

$f$

$O(1)$