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)

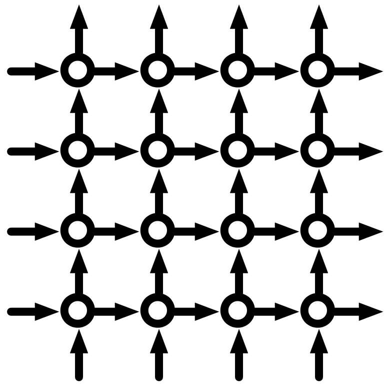
Orr Fischer (Tel Aviv)

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

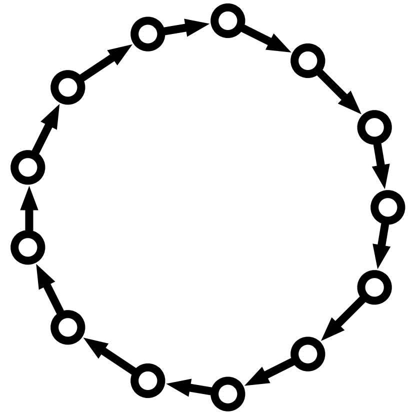
- **LCL = locally checkable labelling**
  - Naor–Stockmeyer (1995)
- Valid solution can be detected by checking $O(1)$-radius neighbourhood of each node
  - maximal independent set, maximal matching, vertex colouring, edge colouring …

LCL problems

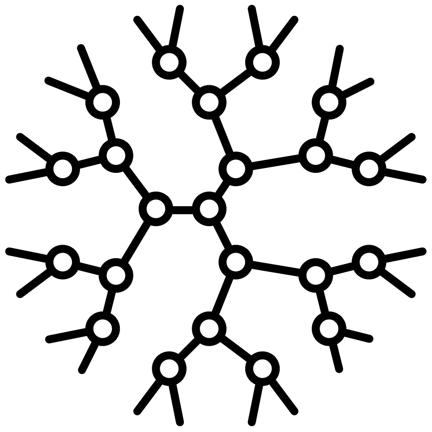
- All LCL problems can be solved with $O(1)$-round ***nondeterministic*** algorithms
  - guess a solution, verify it in $O(1)$ rounds
- Key question: how fast can we solve them with ***deterministic*** algorithms?
  - cf. P vs. NP

Traditional settings

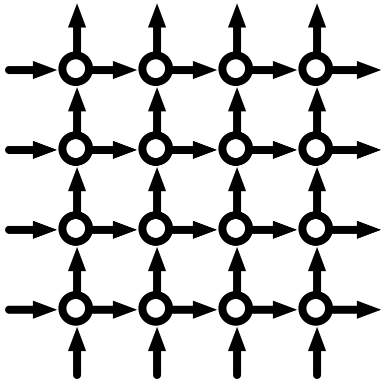
- **Directed cycles**
  - Cole–Vishkin (1986), Linial (1992)…
  - well understood



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

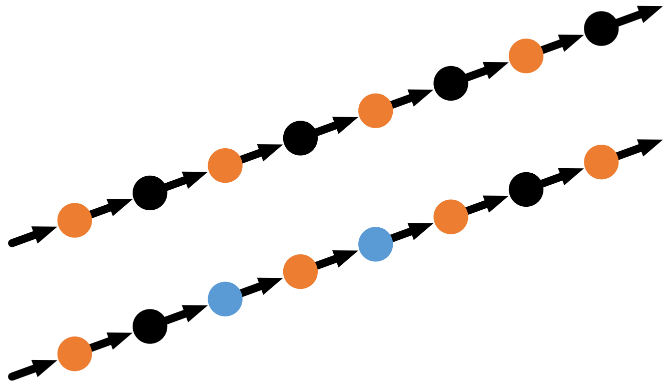


Our setting today



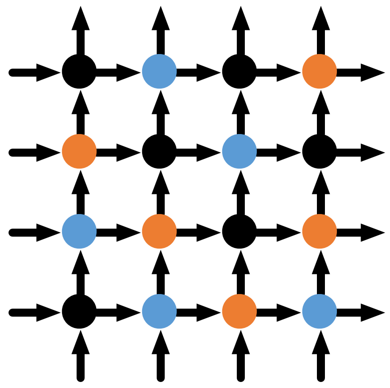
- **Oriented grids** (2D)
  - toroidal grid, $n \times n$ nodes, unique identifiers
  - consistent orientations north/east/south/west
- ***Generalisation of directed cycles*** (1D)
- Closer to real-world systems than expander-like worst-case constructions?

Warm-up examples



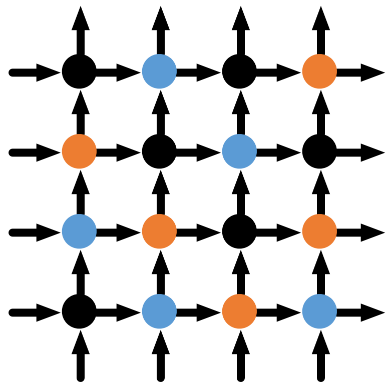
- Vertex colouring in **1D grids**
- **2-colouring:** global, $\Theta(n)$ rounds
- **3-colouring:** local, $\Theta(\log^* n)$ rounds
  - Cole–Vishkin (1986), Linial (1992)

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, **polylog(n)** …
    (Brand et al. 2016)
  - **undecidable** (Naor–Stockmeyer 1995)

Complexity of LCL problems

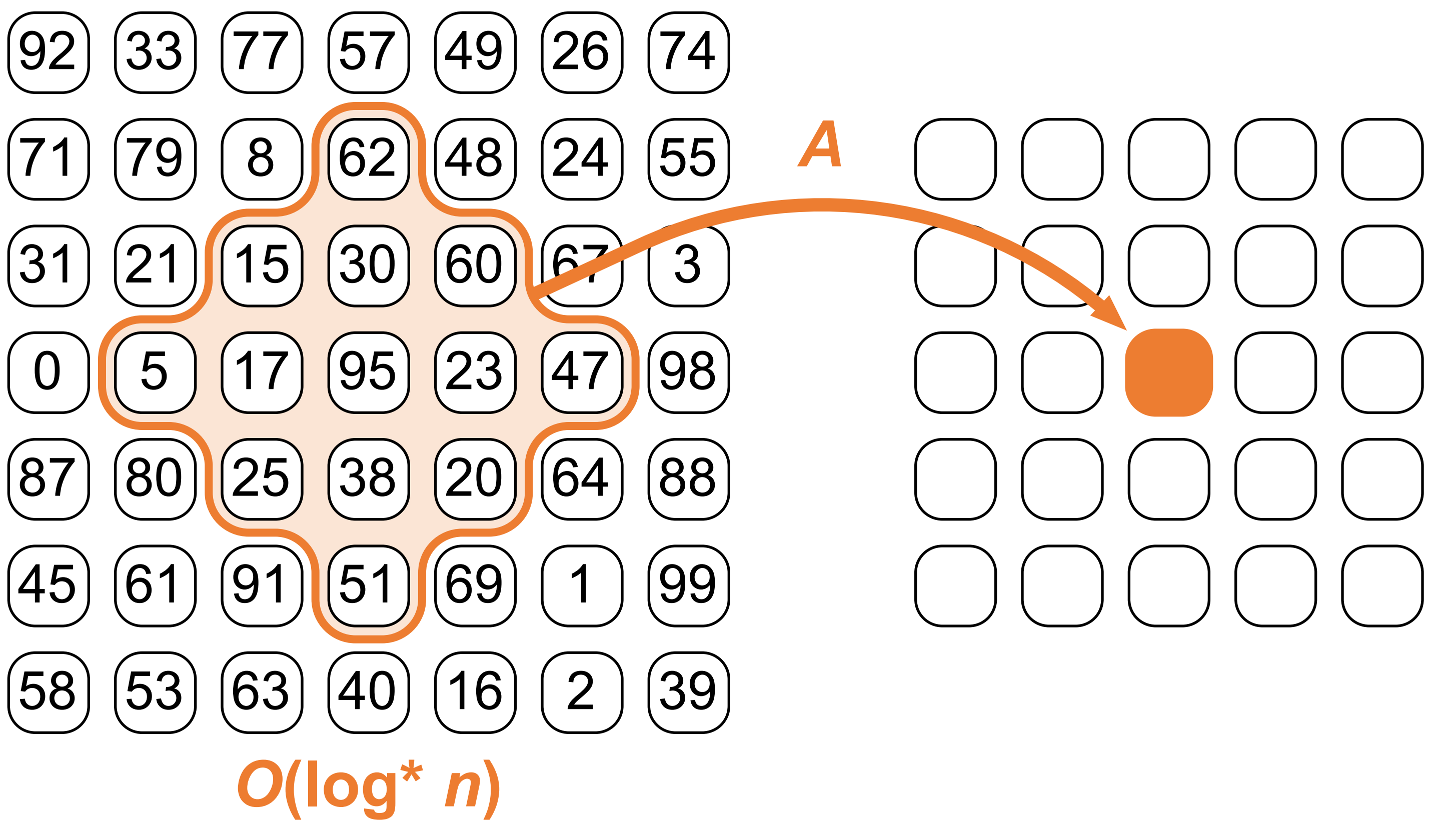
- 1D grids:
  - everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  - **decidable**
- 2D grids:
  - everything is $O(1)$, $\Theta(\log^* n)$, or $\Theta(n)$
  - **undecidable**

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??

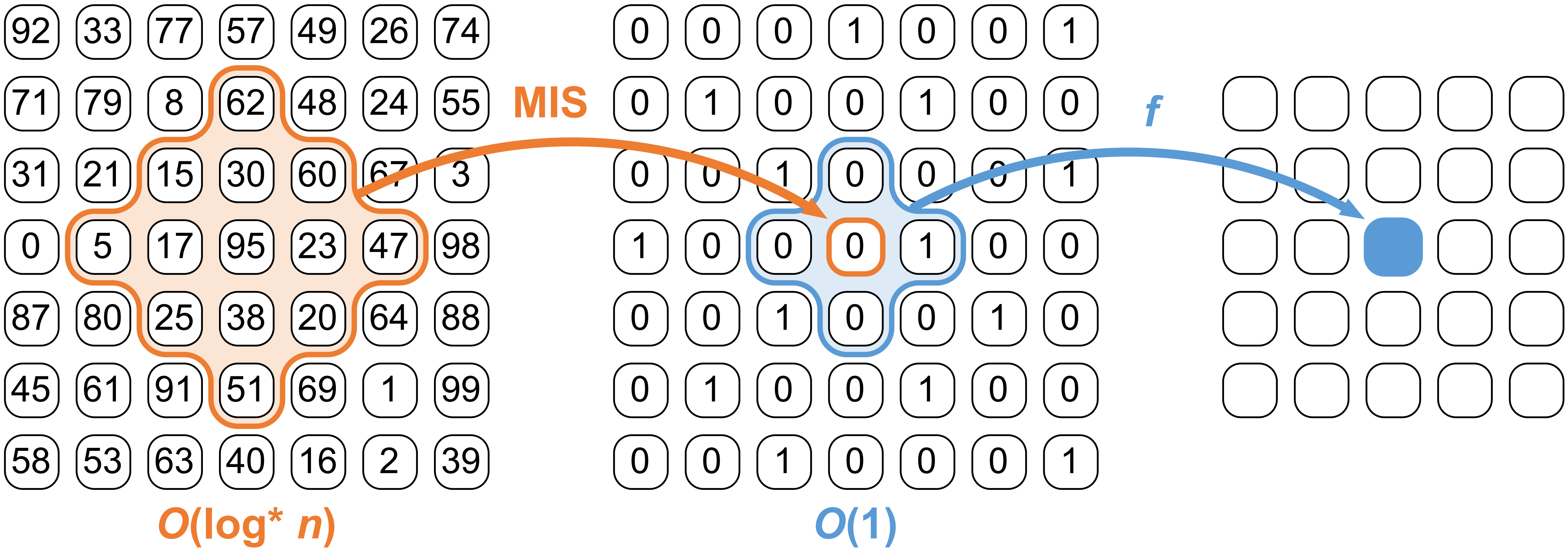

92 33 77 57 49 26 74
71 79 8 62 48 24 55
31 21 15 30 60 67 3
0 5 17 95 23 47 98
87 80 25 38 20 64 88
45 61 91 51 69 1 99
58 53 63 40 16 2 39
A
O(log* n)

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 $\Theta(\log^* n)$-time algorithm can be translated to a "***normal form***"
  - we isolate a fixed $\Theta(\log^* n)$-time component
  - everything else is a ***finite function***


MIS
f
O(log* n)
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 $\boldsymbol{o(n)}$ in $\boldsymbol{n \times n}$ grids
- Solving P in time $\boldsymbol{O(\log^* N)}$ in $\boldsymbol{N \times N}$ grids:
  - pick suitable $n = O(1)$, $k = O(1)$
  - find MIS in G^k
  - use MIS to find ***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 $\boldsymbol{k} = 3$, $\boldsymbol{r} = 7$
- Algorithm ≈ 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

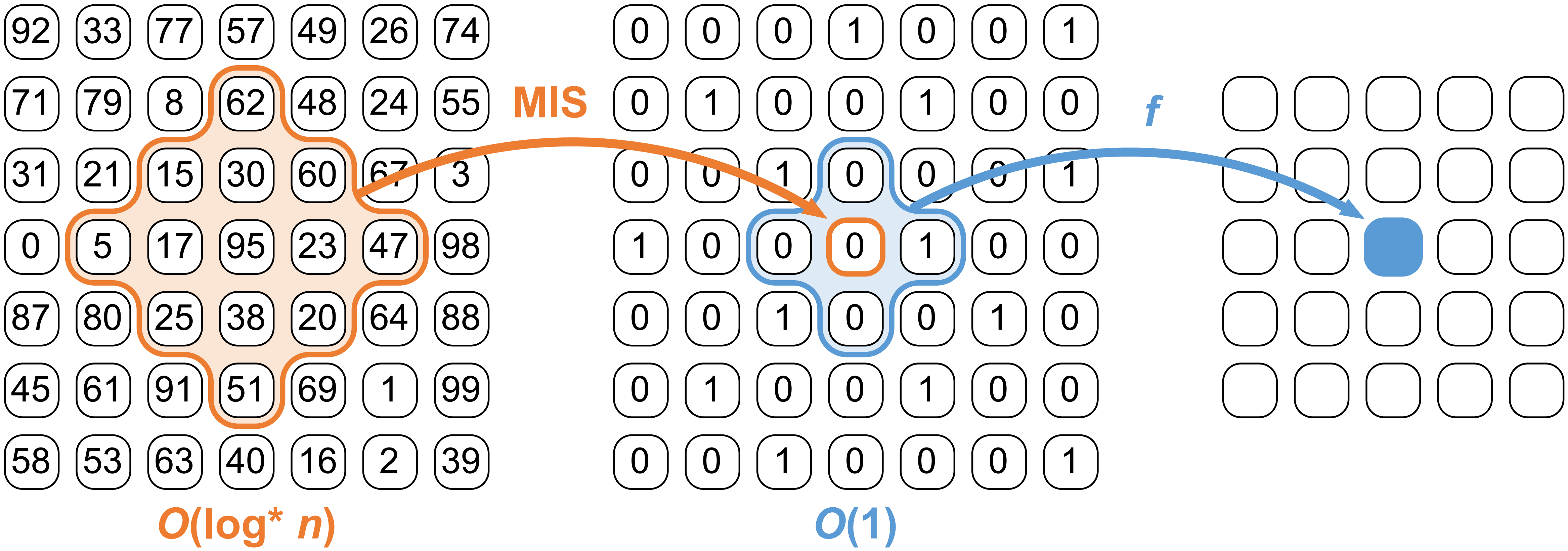
- **Enumerate all $r \times r$ neighbourhoods** that represent possible fragments of maximal independent sets in G^k
- Construct **neighbourhood graphs**
  - algorithm ≈ labelling of neighbourhood graph
- Apply **SAT solvers** to find a labelling

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, but more about this in some other talk!

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***?


92 33 77 57 49 26 74
71 79 8 62 48 24 55
31 21 15 30 60 67 3
0 5 17 95 23 47 98
87 80 25 38 20 64 88
45 61 91 51 69 1 99
58 53 63 40 16 2 39
MIS
0 0 0 1 0 0 1
0 1 0 0 1 0 0
0 0 1 0 0 0 1
1 0 0 0 1 0 0
0 0 1 0 0 1 0
0 1 0 0 1 0 0
0 0 1 0 0 0 1
f
O(log* n)
O(1)