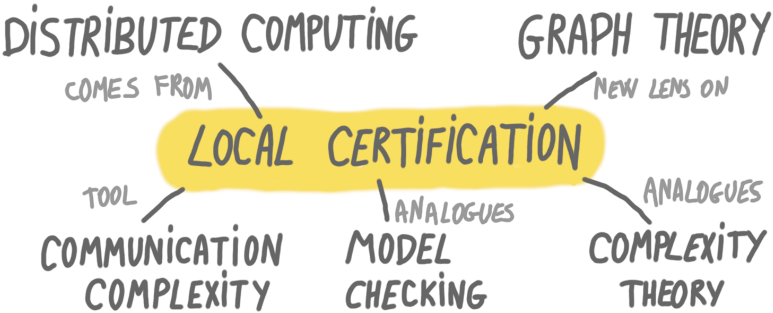
Introduction to local certification

Laurent Feuilloley

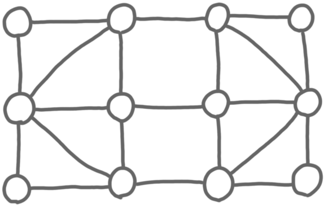
CNRS and University of Lyon · GT Graphes

JNIFM · Grenoble · March 2024

Landscape

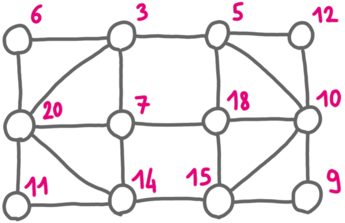

DISTRIBUTED COMPUTING
GRAPH THEORY
COMES FROM
NEW LENS ON
LOCAL CERTIFICATION
TOOL
ANALOGUES
ANALOGUES
COMMUNICATION COMPLEXITY
MODEL CHECKING
COMPLEXITY THEORY

Distributed perspective on graphs



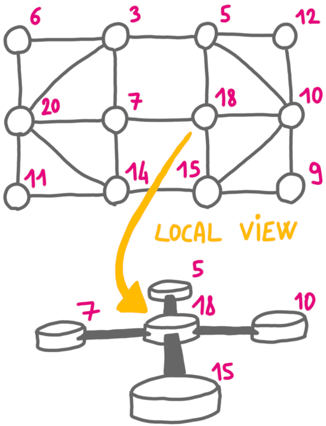
- The graph represents a network.
- Nodes are machines.
- Edges are communication channels.

Distributed perspective on graphs



- The graph represents a network.
- Nodes are machines.
- Edges are communication channels.
- Unique identifiers.

Distributed perspective on graphs



- The graph represents a network.
- Nodes are machines.
- Edges are communication channels.
- Unique identifiers.
- In this talk:
  - Every node sees its neighbors,
  - runs the algorithm,
  - outputs a binary decision.

Local recognition of graph classes

Let $\mathcal{C}$ be a class of connected graphs (*e.g.* planar graphs)

Local recognition of $\mathcal{C}$: A local decision algorithm such that:

- If $G \in \mathcal{C}$ then all the vertices accept.
- If $G \notin \mathcal{C}$ then at least one vertex rejects.

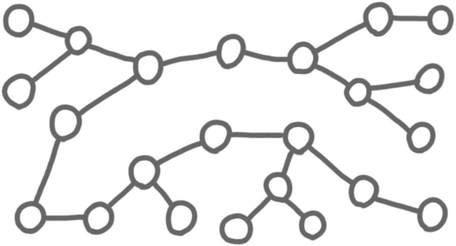
Examples:

- Graphs of degree 3 can be recognized locally.
- For any set $S \subseteq \mathbb{N}$, graph with all degrees in S can be recognized locally.
- Trees cannot be recognized locally

Quick proof for trees

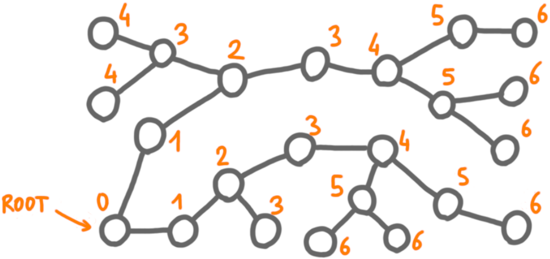
- Consider a graph with a unique cycle that is too large to fit in the view of a node.
- For correctness, a least one node v rejects.
- Now, remove an edge far from v, to cut the cycle. Rerun the checking. Node v still rejects. Contradiction.

Introducing local certification



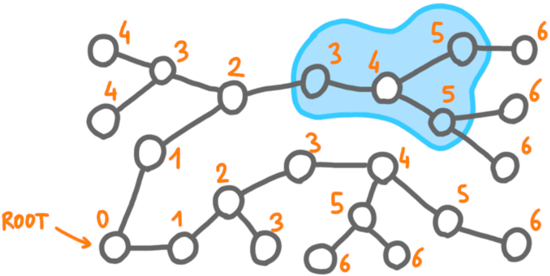
- **Idea:** Use additional information, in the form of labels.

Introducing local certification



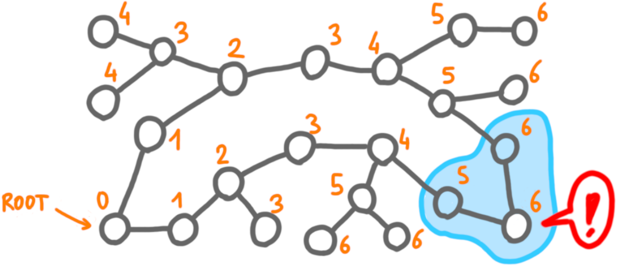
- **Idea:** Use additional information, in the form of labels.
- For trees: the distances from an arbitrarily chosen node.

Introducing local certification



- **Idea:** Use additional information, in the form of labels.
- For trees: the distances from an arbitrarily chosen node.
- Sanity check at each node: the distances locally make sense.

Introducing local certification



- **Idea:** Use additional information, in the form of labels.
- For trees: the distances from an arbitrarily chosen node.
- Sanity check at each node: the distances locally make sense.
- **Key property:** if the graph has a cycle, at least one node will reject.

Definition and story

Definition: A local certification of a graph class $\mathcal{C}$ is a local decision algorithm such that :

1. For $G \in \mathcal{C}$, **there exists** certificate assignment that makes **all vertices** accept.
2. For $G \notin \mathcal{C}$, **for any** certificate assignment, **at least one vertex** rejects.

Story:

- A *prover* is trying to convince all nodes that the graph is in class $\mathcal{C}$ (which might be true or false).
- If the graph is indeed in the class, it succeeds.
- If the graph is not in the class, it cannot succeed.

Any class can be certified

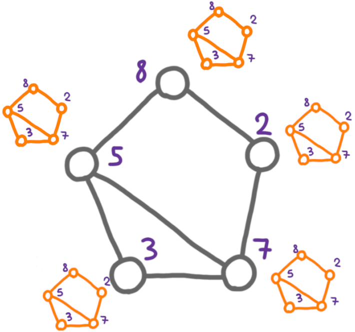
Theorem: Any graph class $\mathcal{C}$ can be certified locally.

When $G \in \mathcal{C}$ the prover gives:

- map of graph with identifiers

Nodes check:

- Same map as neighbors
- Consistent with local view
- Graph given belongs to $\mathcal{C}$.

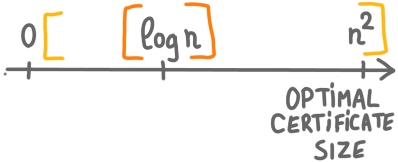


(Unique identifiers are essential here to avoid symmetry issues.)

Certificate size

Question: For a class $\mathcal{C}$, what is the minimum certificate size?

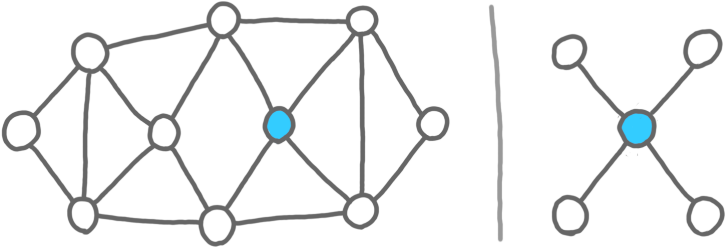
Previous slide $\rightarrow$ Upper bound of $O(n^2)$ bits.



- For some classes $O(1)$ bits suffice.
  $\rightarrow$ degree-3 graphs, 3-colorable graphs.
- For some classes $\Omega(n^2)$ is needed
  $\rightarrow$ Symmetric graphs, non-3-colorable graphs.
- A key size is $\Theta(\log n)$ (aka "compact local certifications").
  $\rightarrow$ Trees, *planar graphs*.

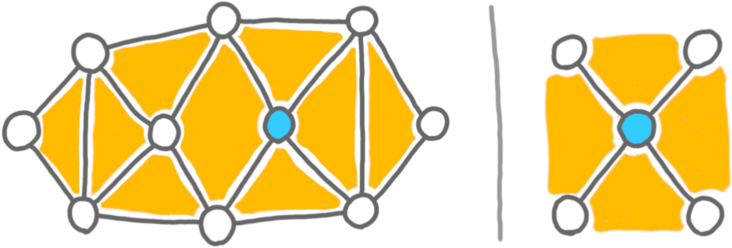
Certifying planarity

Theorem: We can certify planar graphs with $O(\log n)$ bits.

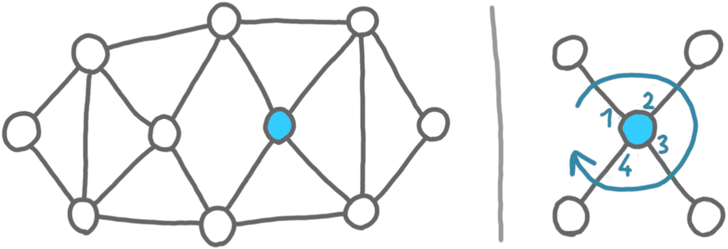


Certifying planarity

Theorem: We can certify planar graphs with $O(\log n)$ bits.

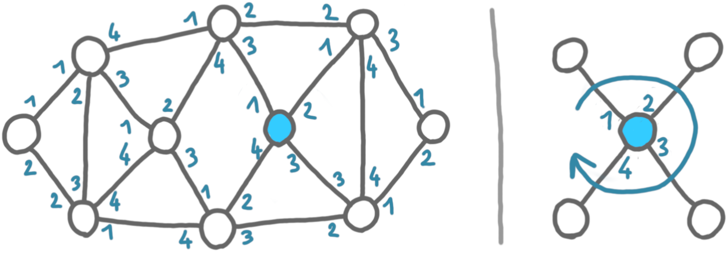


Part 1: Rotational system



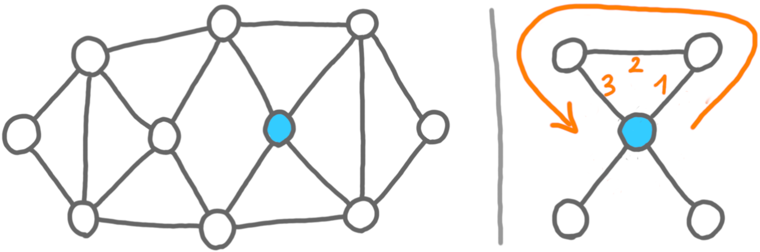
- First component: orientations of the edges around each node.

Part 1: Rotational system



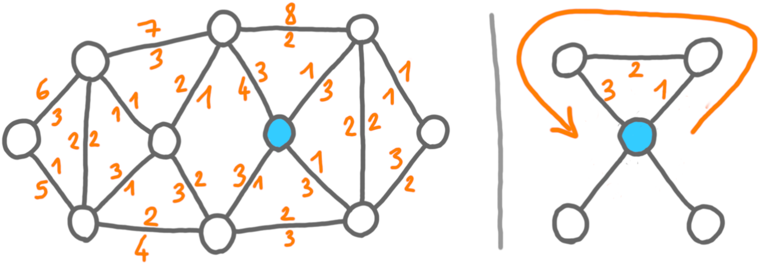
- First component: orientations of the edges around each node.

Part 1: Rotational system



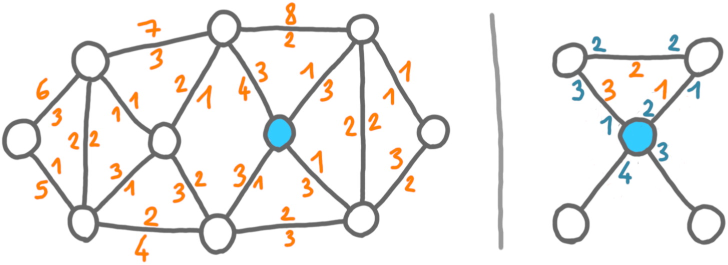
- First component: orientations of the edges around each node.
- Second component: orientations of the edges around each *face*.

Part 1: Rotational system



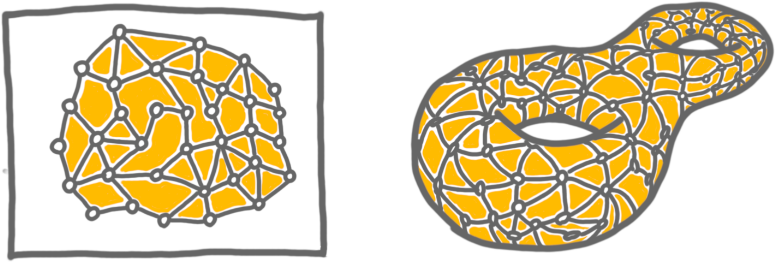
- First component: orientations of the edges around each node.
- Second component: orientations of the edges around each *face*.

Part 1: Rotational system



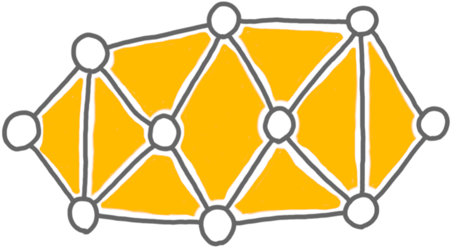
- First component: orientations of the edges around each node.
- Second component: orientations of the edges around each *face*.
- The consistency a system can be checked locally
  $\rightarrow$ Good for certification, if we allow edge certificates.

Part 2: Checking Euler formula



- A rotational system $\rightarrow$ local embedding. Maybe not planar.

Part 2: Checking Euler formula

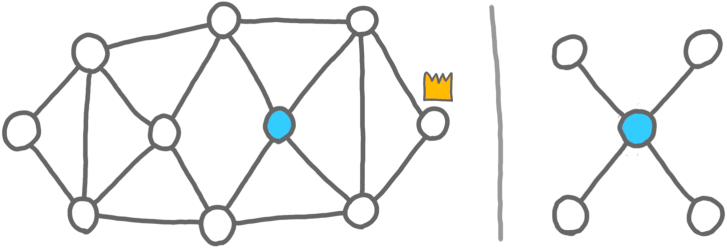


G is planar iff

$|V| - |E| + |F| = 2$

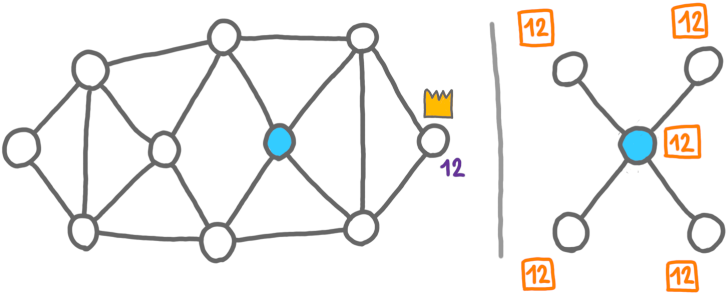
- A rotational system $\rightarrow$ local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.

Part 2: Checking Euler formula



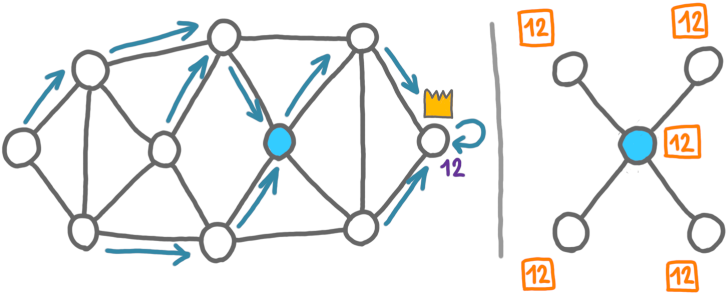
- A rotational system $\rightarrow$ local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.

Part 2: Checking Euler formula



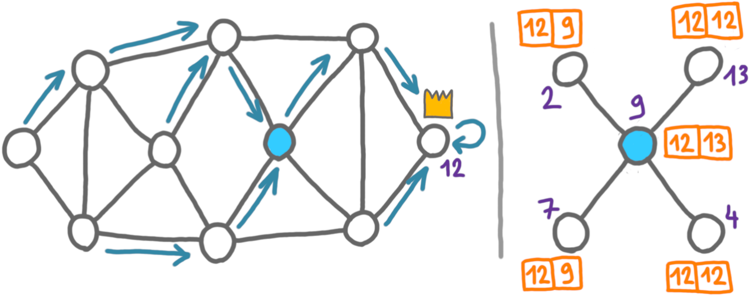
- A rotational system $\rightarrow$ local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID,

Part 2: Checking Euler formula



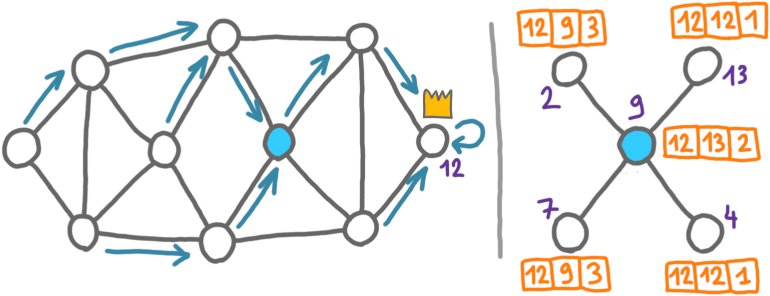
- A rotational system $\rightarrow$ local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID,

Part 2: Checking Euler formula



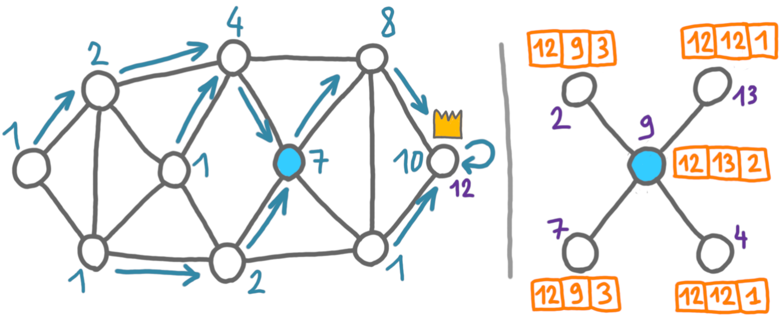
- A rotational system $\rightarrow$ local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID, Parent's ID,

Part 2: Checking Euler formula



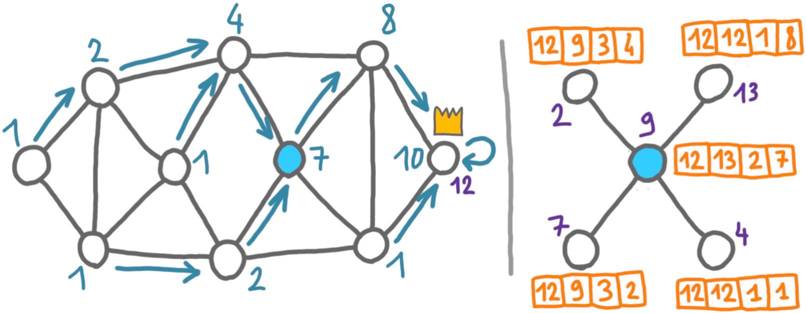
- A rotational system → local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID, Parent's ID, distance to leader

Part 2: Checking Euler formula



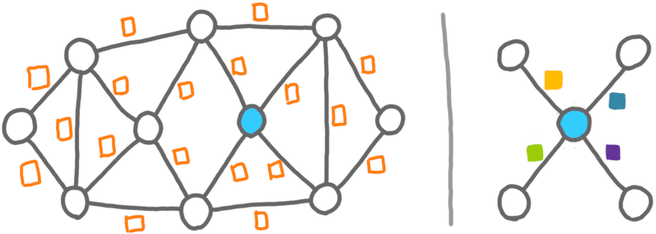
- A rotational system → local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID, Parent's ID, distance to leader
- Euler checking: Use the tree to gather the information.

Part 2: Checking Euler formula



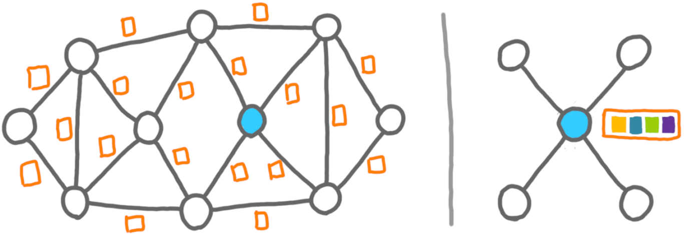
- A rotational system → local embedding. Maybe not planar.
- Euler formula characterizes planar embeddings.
- Plan: Gather the information at a leader node for checking.
- For a leader: Leader's ID, Parent's ID, distance to leader
- Euler checking: Use the tree to gather the information.

Part 3: Getting rid of edge certificates



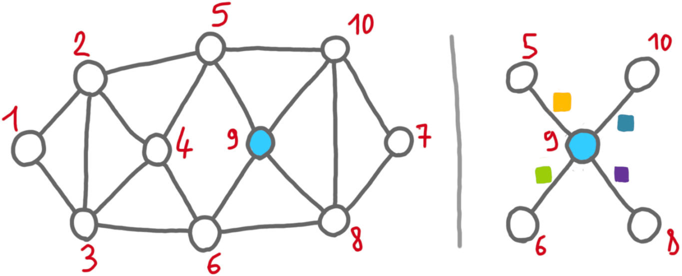
- Rotational systems naturally translate into edge certificates.

Part 3: Getting rid of edge certificates



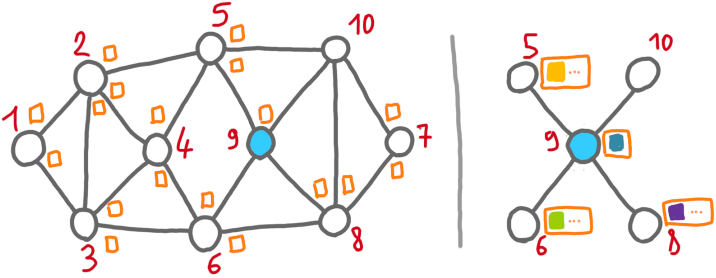
- Rotational systems naturally translate into edge certificates.
- Duplicating on both endpoints $\rightarrow$ size $=$ max-degree $\times O(\log n)$

Part 3: Getting rid of edge certificates



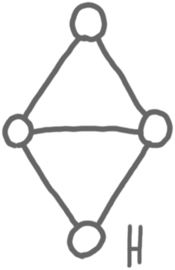
- Rotational systems naturally translate into edge certificates.
- Duplicating on both endpoints $\rightarrow$ size $=$ max-degree $\times O(\log n)$
- Planar graphs have degeneracy ≤ 6.

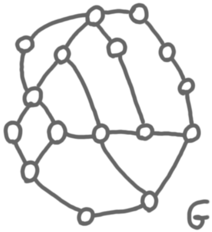
Part 3: Getting rid of edge certificates



- Rotational systems naturally translate into edge certificates.
- Duplicating on both endpoints $\rightarrow$ size $=$ max-degree $\times O(\log n)$
- Planar graphs have degeneracy ≤ 6.
- Each edge certificate goes to the smallest-index vertex.
  $\rightarrow$ Ok for the verification phase, and $O(\log n)$ certificates!

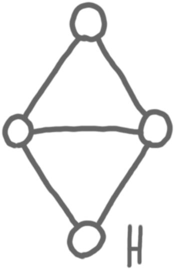
Forbidden minors

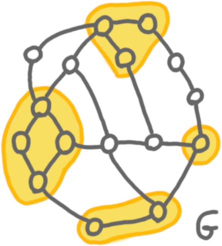




- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.

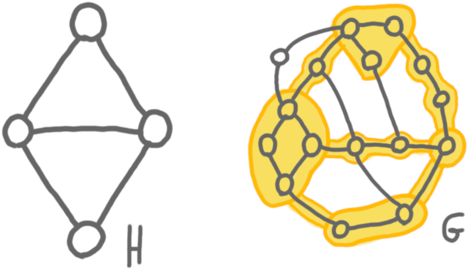
Forbidden minors





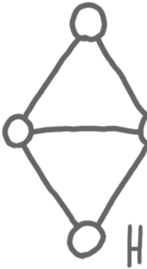
- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.

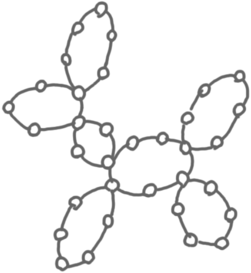
Forbidden minors



- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.

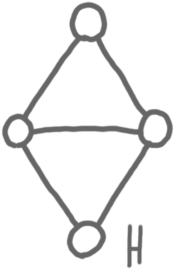
Forbidden minors

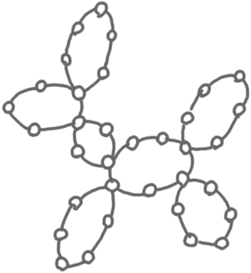




- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.

Forbidden minors



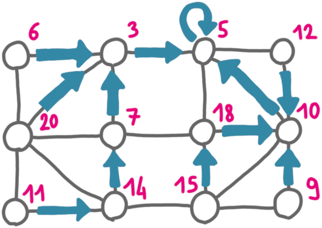


- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.
- **Open question:** Is it true that every class defined by excluded minors can be certified with $O(\log n)$ bits?

Forbidden minors

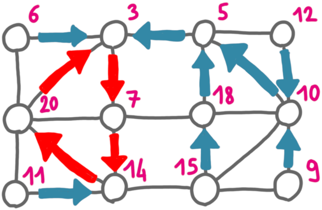
- Planar graphs = graphs that do not have $K_5, K_{3,3}$ as minors.
- **Open question:** Is it true that every class defined by excluded minors can be certified with $O(\log n)$ bits?
- Known true for bounded-genus, small minors, planar minors.

Origin: distributed computing



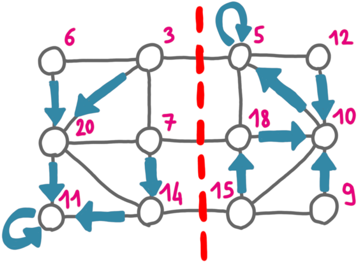
- A classic problem: Compute a spanning tree.

Origin: distributed computing



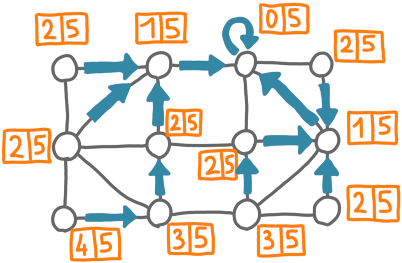
- A classic problem: Compute a spanning tree.
- Fault-tolerance: be able to detect locally if the tree is broken.

Origin: distributed computing



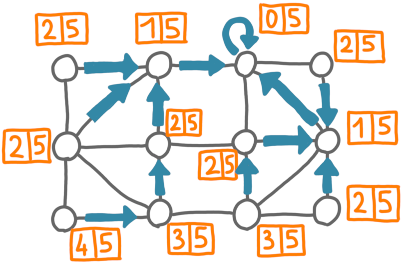
- A classic problem: Compute a spanning tree.
- Fault-tolerance: be able to detect locally if the tree is broken.

Origin: distributed computing



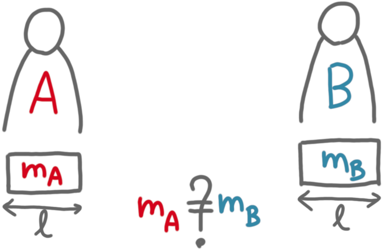
- A classic problem: Compute a spanning tree.
- Fault-tolerance: be able to detect locally if the tree is broken.
- Impossible if only the pointers are kept in memory, but possible if one also keeps ID of and distance to the root.

Origin: distributed computing



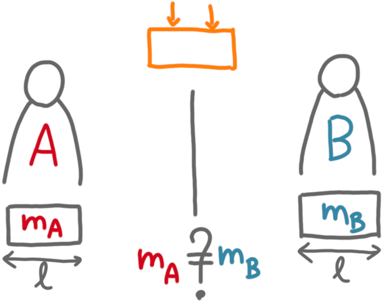
- A classic problem: Compute a spanning tree.
- Fault-tolerance: be able to detect locally if the tree is broken.
- Impossible if only the pointers are kept in memory, but possible if one also keeps ID of and distance to the root.
- Algorithms that can cope with such faults are called *self-stabilizing*.

Tool: Communication complexity



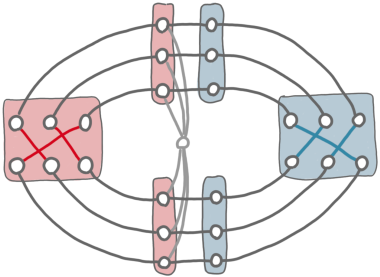
- How much communication to compute a function, when the input is distributed among several players.

Tool: Communication complexity



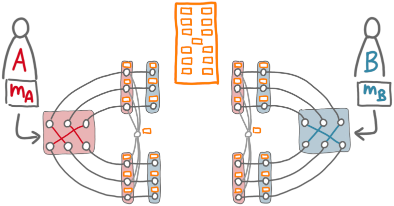
- How much communication to compute a function, when the input is distributed among several players. (Non-deterministic variant.)

Tool: Communication complexity



- How much communication to compute a function, when the input is distributed among several players. (Non-deterministic variant.)
- Tailored instances for reductions.

Tool: Communication complexity



- How much communication to compute a function, when the input is distributed among several players. (Non-deterministic variant.)
- Tailored instances for reductions.
- Each player gets one part of the graph. Argue about certificate size at boundary.

Analogy: Complexity theory

Class NP:

There exists a polytime verification algorithm A such that:

Input is correct	$\Leftrightarrow$	Exists c such that $A(c, \mathit{input})$ accepts.

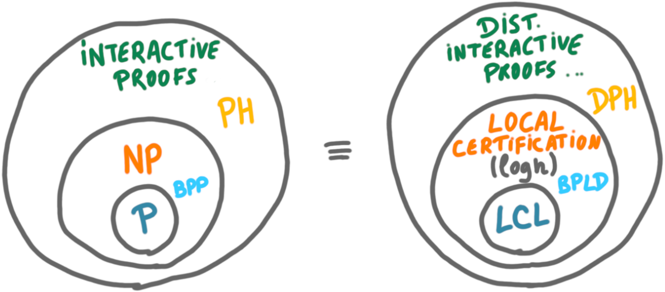
Local certification:

There exists a local sanity check A such that:

Input is correct	$\Leftrightarrow$	Exists $c : V \to$ labels such that $A(c, \mathit{input})$ accepts at every node.

Analogy: Complexity theory

One can define many other analogues of the classic complexity classes: probabilistic classes, interactive proofs, zero-knowledge, polynomial hierarchy etc.



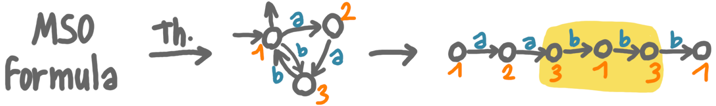
Analogy: Model checking

General model checking approach: Check efficiently that some restricted properties on restricted structures.

Courcelle theorem: Any MSO formula can be checked in polynomial-time in graphs of bounded treewidth.

Recent analogues: Any MSO formula can be certified with $O((poly)\log n)$ bits in graphs on graphs of bounded treedepth/treewidth/cliquewidth.

Techniques: Kernelization, automata theory.



Wrapping up

- Local certification is about checking locally a graph property (or data structure), thanks to certificates.
- It originates from the study of fault-tolerance in distributed computing.
- It is connected to several other areas of TCS.
- There are still exciting open questions!

Wrapping up

- Local certification is about checking locally a graph property (or data structure), thanks to certificates.
- It originates from the study of fault-tolerance in distributed computing.
- It is connected to several other areas of TCS.
- There are still exciting open questions!

Thanks for your attention!

(I cannot be around for the rest of the week, do not hesitate to send me an email for additional questions!)