Flinders Hamiltonian Cycle Problem Challenge

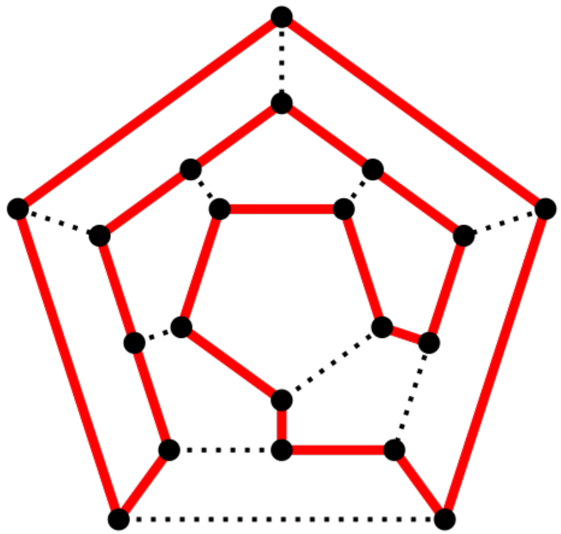
Nathann Cohen

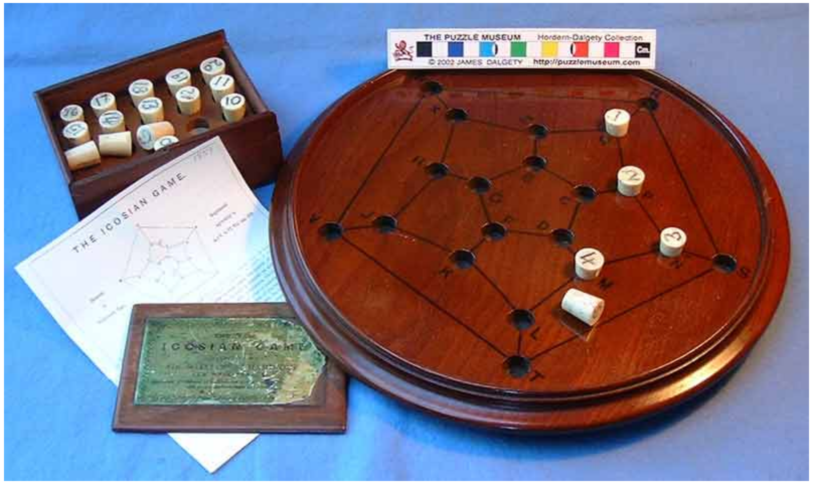
CNRS, LRI

David Coudert

COATI

Hamiltonian Cycle -- HCP





Hamilton, 1857

Graph theory (Kirkman, 1856 / Hamilton 1857):

- Given G = (V, E), find a **cycle** that visits **each vertex** once and only once
- **NP-complete** (decision problem)
- Central in graph theory, used for proving NP-completeness
- Generalization: **traveling salesman problem** (optimization problem)

What is known

Hundreds of papers, many algorithms

TSP-based softwares

- **Concorde** – good at handling structure, branch-and-bound
- **LKH** – very fast for very large… road maps

Specialized HCP

- Snakes and Ladders heuristics
- Chalaturnyk's algorithm – efficient only on very sparse graphs

The challenge

Organizer

- **Michael Haythorpe**, Flinders Hamiltonian Cycle Project (Flinders U., Adelaide)
- Design algorithms and **hard instances** for Hamiltonian cycle problem
- **Goal**: design benchmark of hard instances

Data set

- **1001 graphs**, from 66 to 9528 nodes, **avg ≈ 3098** — **Huge**
- Hamiltonian by construction
- Designed to be hard to solve using existing algorithms / softwares — **He tried**

Rules

- Competition open from Sept. 30, 2015 till Sept. 30, 2016 (1 year)
- **All methods allowed**, no restriction, no need to provide code
- The first team to give largest number of solutions wins ! — **It's a race**

Results

FHCP 864 ↓

- 12 "serious" answers: **1st – 985** (14/12/15) 2nd – 614 (IBM) avg – 338

Toolbox

Sagemath

- Open-source
- Python based + Cython, C
- Many **graph algorithms**
- Interface for **ILP** solvers
- **Visualization** tools

✚

CPLEX Optimizer

- IBM Ilog
- Powerful ILP solver

✚

Super computer

External libraries used with Sagemath

- **nauty** → graph isomorphisms
- **bliss** → canonical labeling
- **d3.js** → graph visualization (javascript)

We couldn't done it without it

and also an Excel sheet

Methodology

Exploration of graphs

- Measure standard metrics: diameter, min & max degree, etc.
- Use **visualization tool**

Goal

- Identify families of similar graphs (manual classification)
- Identify (sub)-structure to use for the design of algorithms

Design specific methods

- **Separation**, decomposition, small cuts
- **Substitution** of patterns with smaller ones
- Force / discard some edges
- Add *good* constraints to ILP
- Test isomorphism with already solved instances
- …

Reduce the size of instances

Strengthen formulation

… and don't (always) try to understand how we solved an instance

Outline

- **ILP formulation**
- Quick exploration of the graphs
- Resolution methods & tricks
- Unsolved graphs

ILP formulation

Hamiltonian Cycle

- Cycle that visits each vertex once and only once

Predicates

1. Each vertex incident to 2 selected edges (**degree 2**)
2. Set of selected edges is **connected**

Variables

- **b(e)** = 1 if edge e is selected, 0 otherwise (binary)

Constraints on the degree

$$\sum_{v \in N(u)} b(uv) = 2 \qquad \forall u \in V \quad (1)$$

Constraints for connectivity

- Spanning tree, flow, cuts, maximum average degree, etc.

ILP formulation -- flow

Variables **b(e)** = 1 if edge e is selected, 0 otherwise (binary)

f(u,v) ∈ [0,1] flow variable *per arc* (fractional)

Predicates **Degree 2** → each vertex incident to 2 selected edges

Connected → spanning tree, **flow**, cuts, maximum average degree, etc.

$$\sum_{v \in N(u)} b(uv) = 2 \qquad \forall u \in V$$

$$\sum_{v \in N^-(u)} f(v,u) - \sum_{v \in N^+(u)} f(u,v) = \begin{cases} 1 & \text{if } u == s \\ \frac{1}{n-1} & \text{otherwise} \end{cases} \qquad \forall u \in V$$

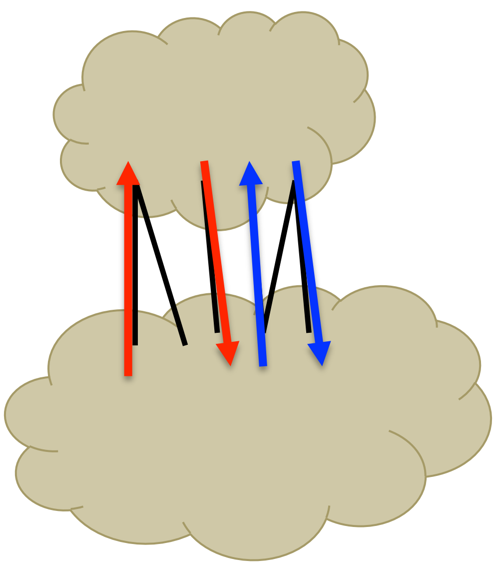
$$f(u,v) \leq b(uv) \qquad \forall u \in V,\ v \in N(u)$$

ILP formulation -- cuts

Variables **b(e)** = 1 if edge e is selected, 0 otherwise (binary)

Predicates **Degree 2** $\rightarrow$ each vertex incident to 2 selected edges

Connected $\rightarrow$ spanning tree, flow, **cuts**, maximum average degree, etc.



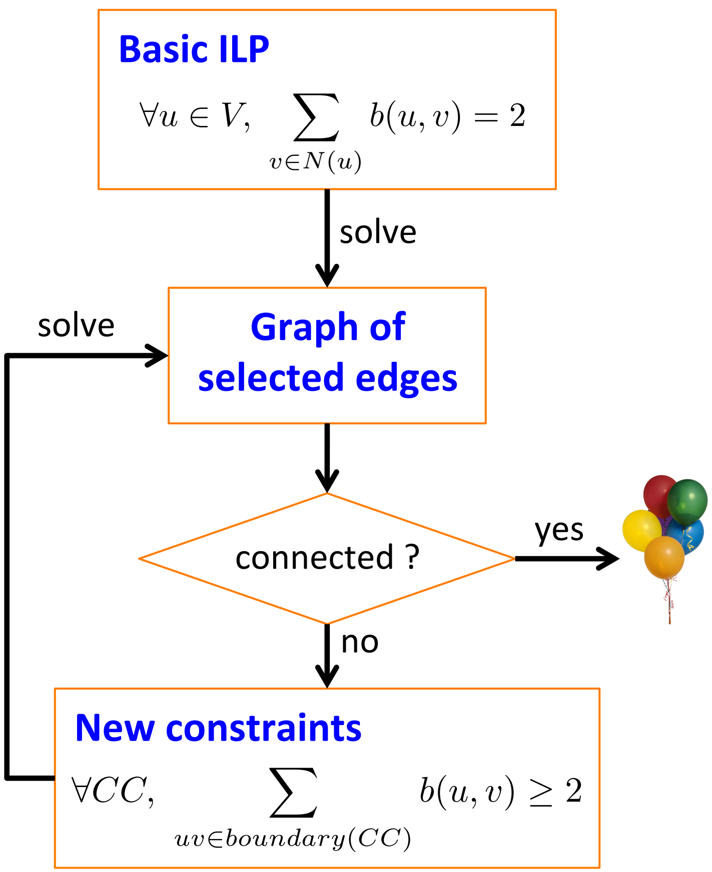
$$\sum_{v \in N(u)} b(uv) = 2 \quad \forall u \in V$$

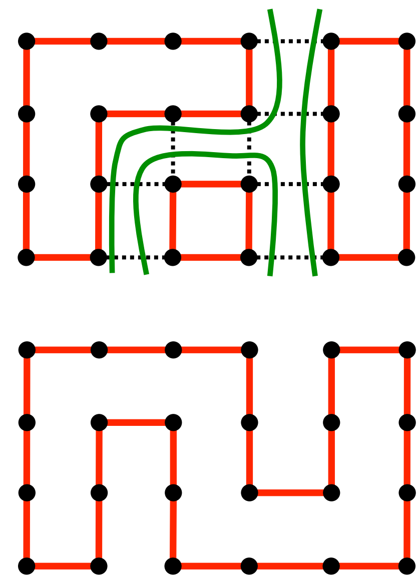
$$\sum_{uv \in C} b(uv) \geq 2 \quad \forall C \in \mathcal{C}$$

sum is even

Exponential number of cuts

ILP formulation -- cuts





Warning: may generate a **huge** number of constraints
$\rightarrow$ need for reducing instance size

Running time

id	n	m	Time (sec)	Gurobi	Other PC
1	66	99	4.64	0.53	0.76
2	70	106	0.01	0.04	0.01
3	78	117	7.17	0.62	0.06
4	84	127	2.50	6.28	10.85
5	90	135	31.71	11.45	33.20
6	94	142	0.74	0.24	0.66
7	102	153	152.18	2.69	42.00
8	108	163	2.42	0.45	1574.00
9	114	171	180.17	69.88	105.30
10	118	178	0.15	0.11	0.17
11	126	189	302.15	48.92	168.10
12	132	199	259.13	1.72	89631.00
...	...	...	...		
1000	**9058**	**13077**	**254.00**	**725.00**	**320.00**

Outline

- ✓ ILP formulation
- **Quick exploration of the graphs**
- Resolution methods & tricks
- Unsolved graphs

Use basic graph properties

Degree 2 $\longrightarrow$ 1908

Diameter 2 $\longrightarrow$ 1113

Min degree	Max degree	Diameter	# graphs
2	399 – 1122	2	11
2	124 – 1908	6	110
2	90 – 392	10	60
3	3	10 – 1113	87
3	4	11 – 1084	300
4	14	15 – 111	50

Use visualization tools

d3.js - https://d3js.org/

- JavaScript library for manipulating documents based on data
- Force-directed graph layout
  - Vertices → charged particles → repulsion
  - Edges → springs → attraction

Available from Sagemath

- Thank you Nathann !

```
sage: G = read_hcp(62)
sage: G.order(), G.size(), G.diameter(), min(G.degree()), max(G.degree())
(408, 936, 15, 4, 14)
sage: G.show(method='js')
sage:
```

Chrome recommended

Outline

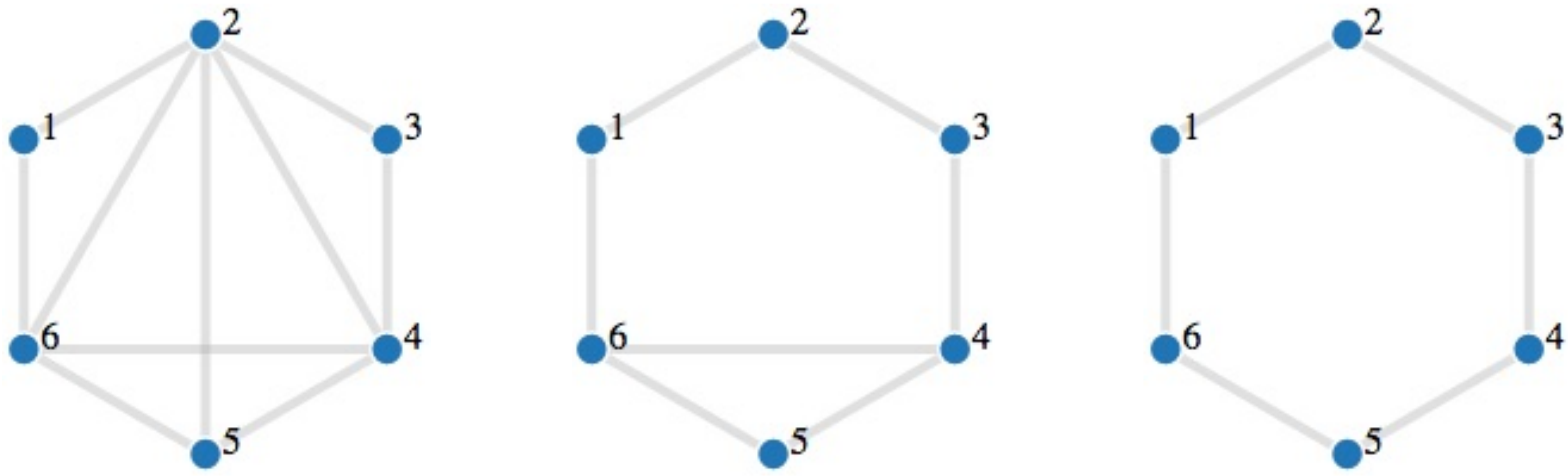
- ✓ ILP formulation
- ✓ Quick exploration of the graphs
- **Resolution methods & tricks**
- Unsolved graphs

Reduction rule: 2 neighbors of degree 2

Observations

- if u has degree 2 → incident edges in the hamiltonian cycle
- if u has **2 neighbors of degree 2** (x and y)

  → **remove other incident edges** (not incident to x or y)



Graphs with diameter 2

id	n	m
59	400	40001
72	460	52901
79	480	57601
84	500	62501
90	510	65026
96	540	72901
128	677	114583
134	724	131045
150	823	169333
162	909	206571
188	1123	315283

- 11 dense graphs
- max degree n-1
- **2 vertices of degree 2 incident to a same vertex**

```
sage: G = read_hcp(188)
sage: G.order(), G.size(), G.diameter(), min(G.degree()), max(G.degree())
(1123, 315283, 2, 2, 1122)
sage:
sage: remove_useless_edges(G)
True
sage: G.is_isomorphic( graphs.CycleGraph( G.order() ) )
True
sage:
```

Graphs with a **unique hamiltonian cycle**
and **largest possible number of edges** [Sheehan, JGT 77]

- $m = n^2/4 + 1$

Try to trick some heuristics due to large number of edges

Graphs with diameter 2

ILP solver uses propagation only

- No addition of cut constraint

$$\forall u \in V, \quad \sum_{v \in N(u)} b(u,v) = 2$$

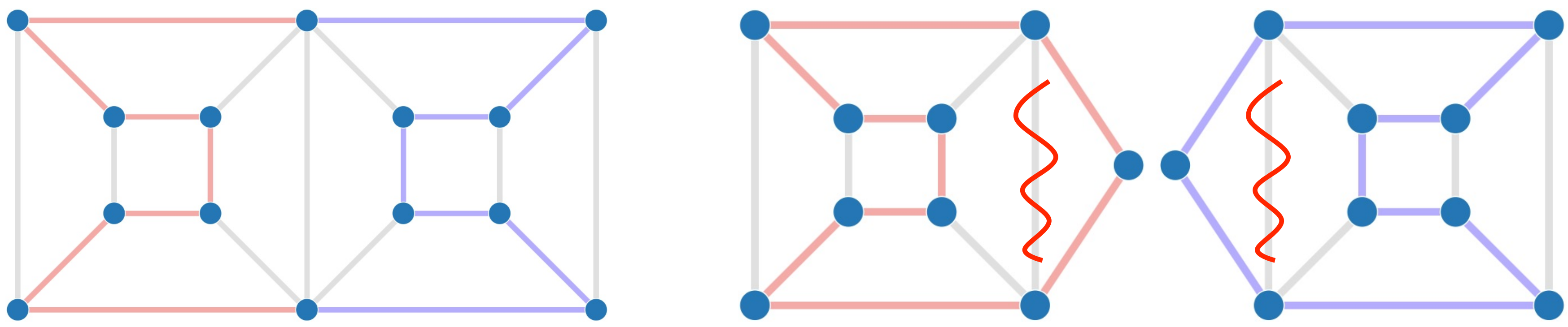
```
sage: G = read_hcp(188)
sage: G.order(), G.size(), G.diameter(), min(G.degree()), max(G.degree())
(1123, 315283, 2, 2, 1122)
sage:
sage: %time tour = hamiltonian_cycle(G, verbose=1, constraint_verbose=1)
Tried aggregator 1 time.
MIP Presolve eliminated 1123 rows and 315283 columns.
All rows and columns eliminated.
Presolve time = 18.21 sec. (2523.37 ticks)

Root node processing (before b&c):
  Real time             =   18.32 sec. (2561.01 ticks)
Parallel b&c, 4 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =   18.32 sec. (2561.01 ticks)
CPU times: user 24.2 s, sys: 1.09 s, total: 25.3 s
Wall time: 25.6 s
sage:
```

Reduction rule: 2-edge/vertex-cut

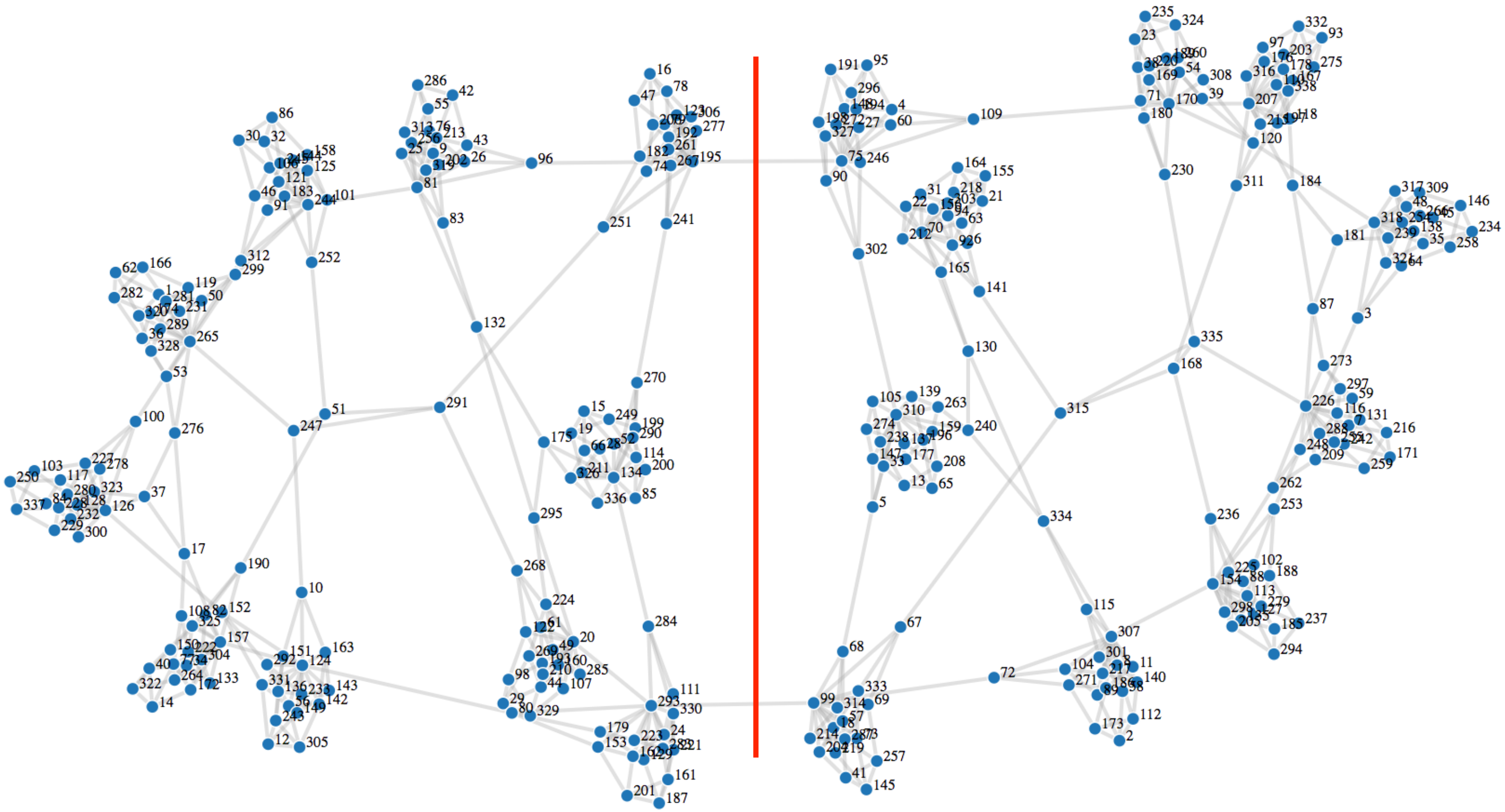
Dividing a graph into triconnected components & SPQR-trees

- Linear time algorithm
- [Hopcroft & Tarjan 73; Gutwenger & Mutzel 01]



#48: 338 nodes

ILP >> 12h computations



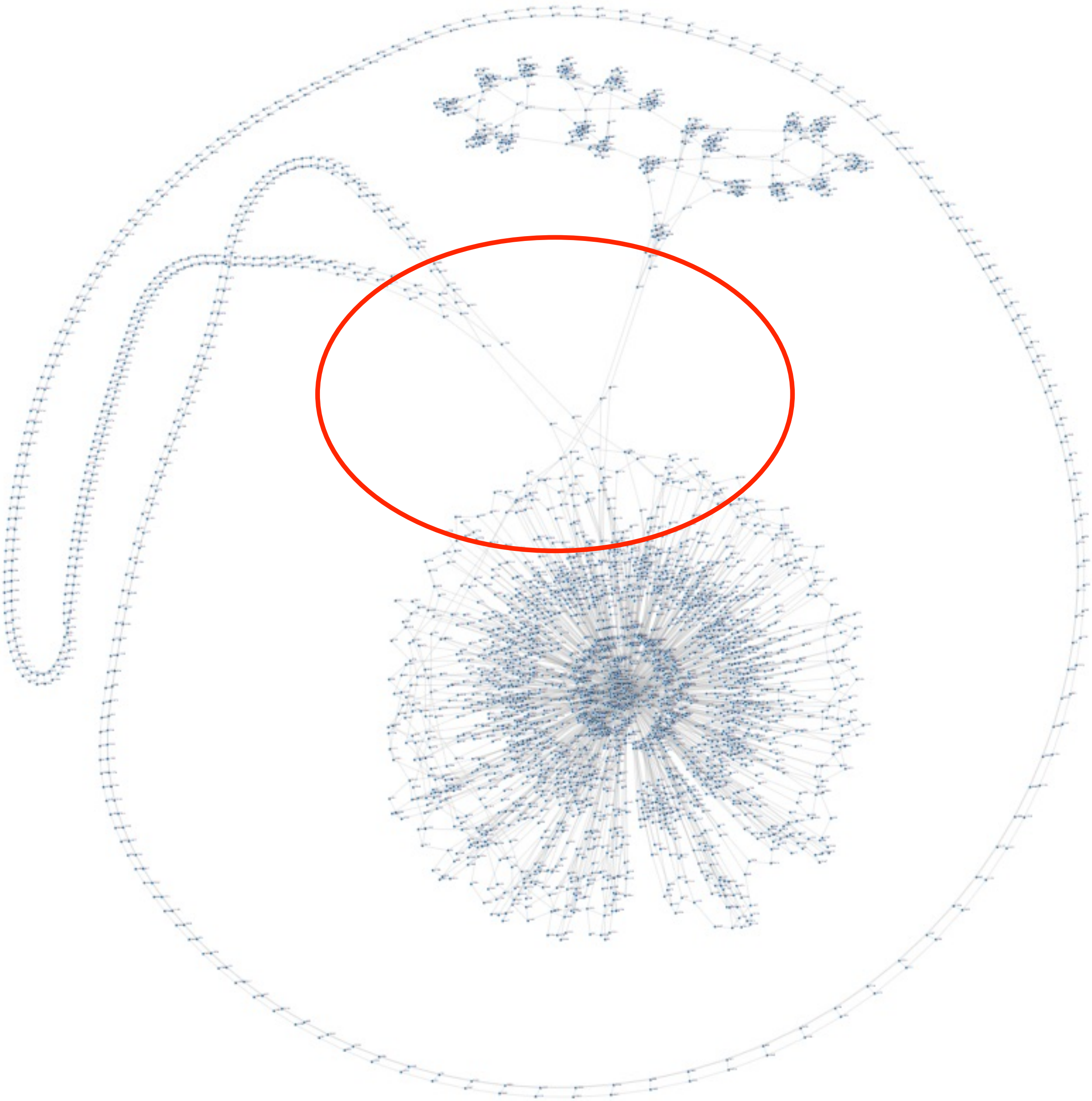
2 edge-cuts
+ isomorphic blocs

After splitting(s): **2 minutes**

555

n = 3273

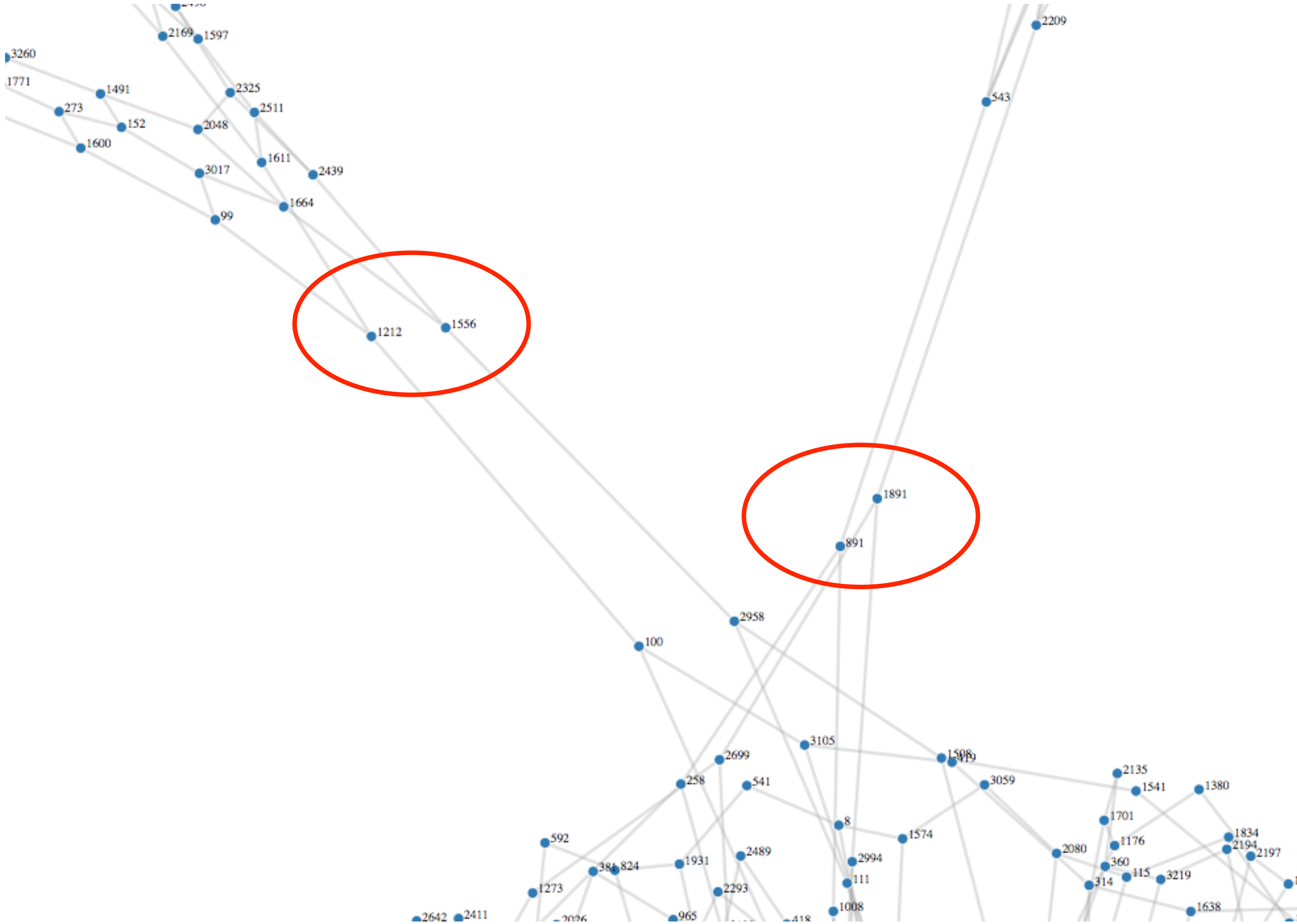
m = 5613



555

n = 3273

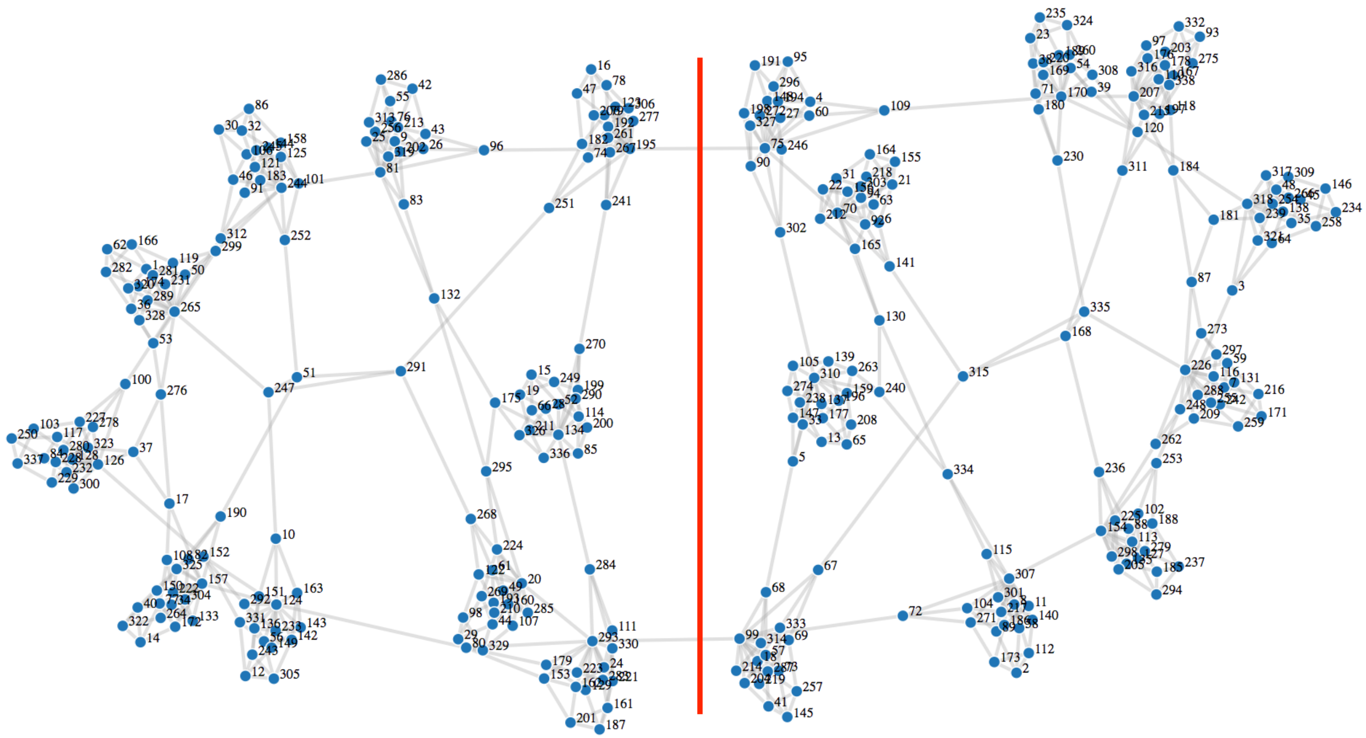
m = 5613



Graph isomorphism

G *isomorphic* to H if there is an *edge-preserving bijection* f: V(G) → V(H)

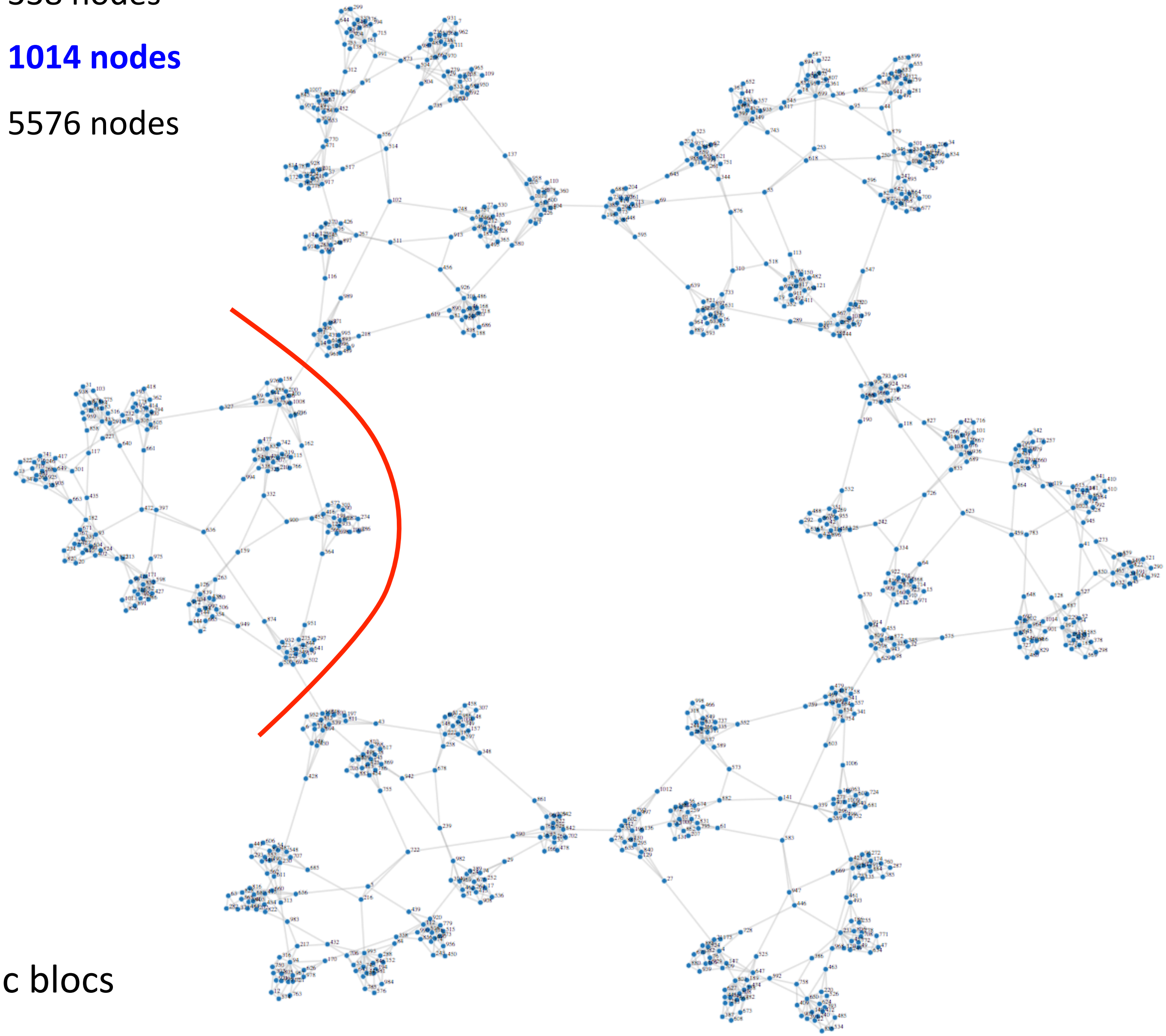
(i.e., edge f(u)f(v) in E(H) if and only if uv in E(G))



nauty http://pallini.di.uniroma1.it/ [McKay 84; McKay & Piperno 14]

- Very fast
- Return the mapping

#48: 338 nodes
…
#175: 1014 nodes
…
#875: 5576 nodes



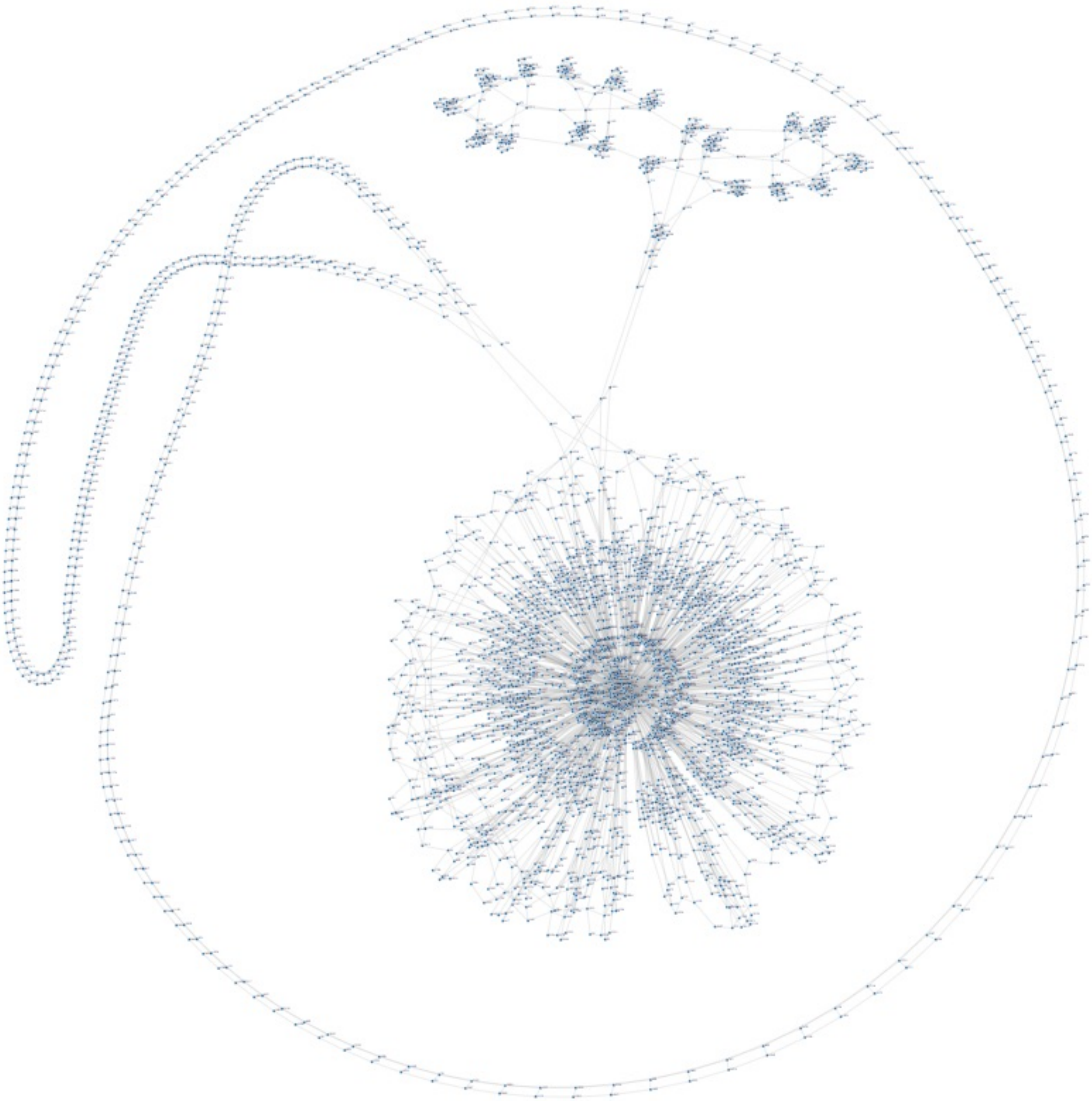
2 edge-cuts
+ isomorphic blocs

COATI

555

$n = 3273$

$m = 5613$



Graph isomorphism -- Trick

Idea: use database of solutions

- Avoid solving twice same instance (from challenge, subgraphs, etc.)

dictionary: graph → hamiltonian cycle

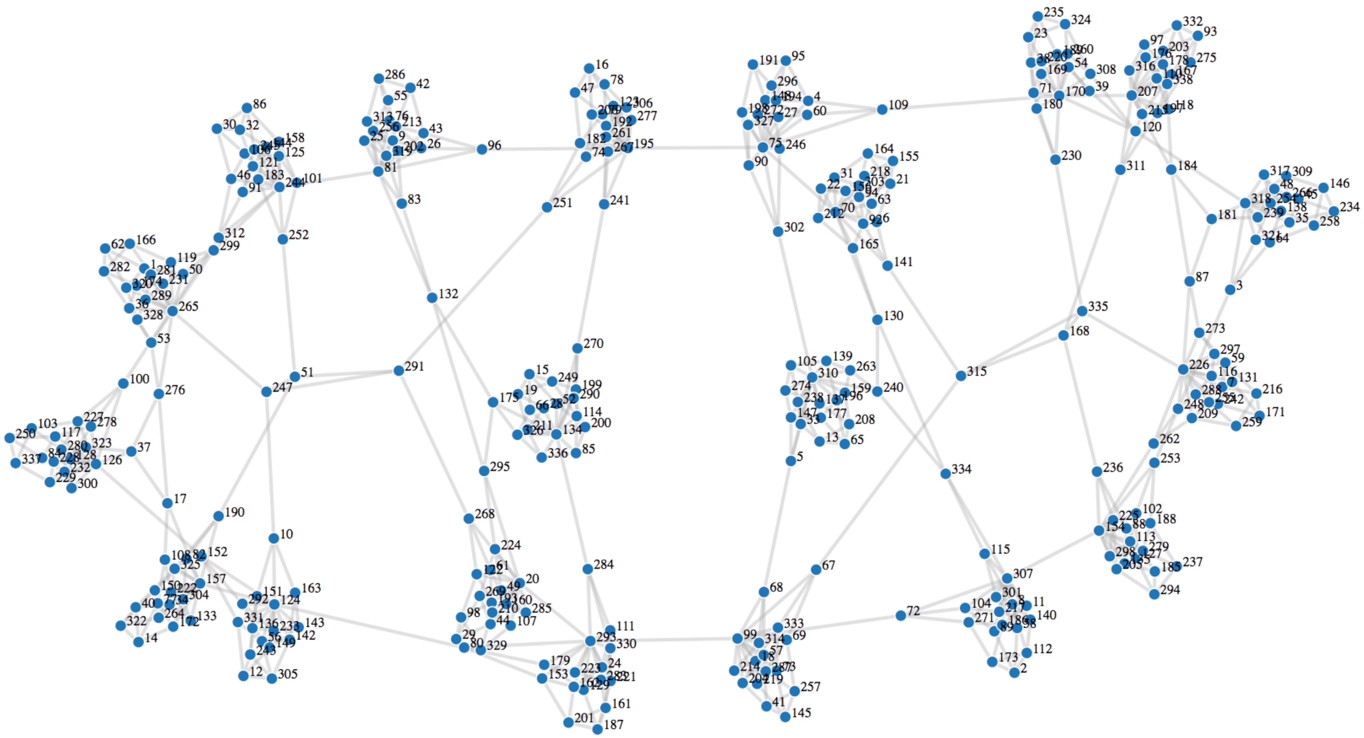
- Canonical labeling = labeling invariant to isomorphism class

  → **bliss** [Junttila & Ksaki 07] http://www.tcs.hut.fi/Software/bliss/
- Immutable graphs are hashable in Sagemath

```
sage: G = graphs.Grid2dGraph(3,3)
sage: H = G.relabel(inplace=False)
sage: print G.vertices()
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
sage: print H.vertices()
[0, 1, 2, 3, 4, 5, 6, 7, 8]
sage:
sage: GC = G.canonical_label()
sage: DB = {GC.copy(immutable=True): 123}
sage:
sage: H.canonical_label().copy(immutable=True) in DB
True
sage:
```

Substitution

#48: **338 nodes**
#62: 408 nodes
…
#175: 1014 nodes
…
#876: 5577 nodes



Many identical blocs

Substitution

#48: 338 nodes

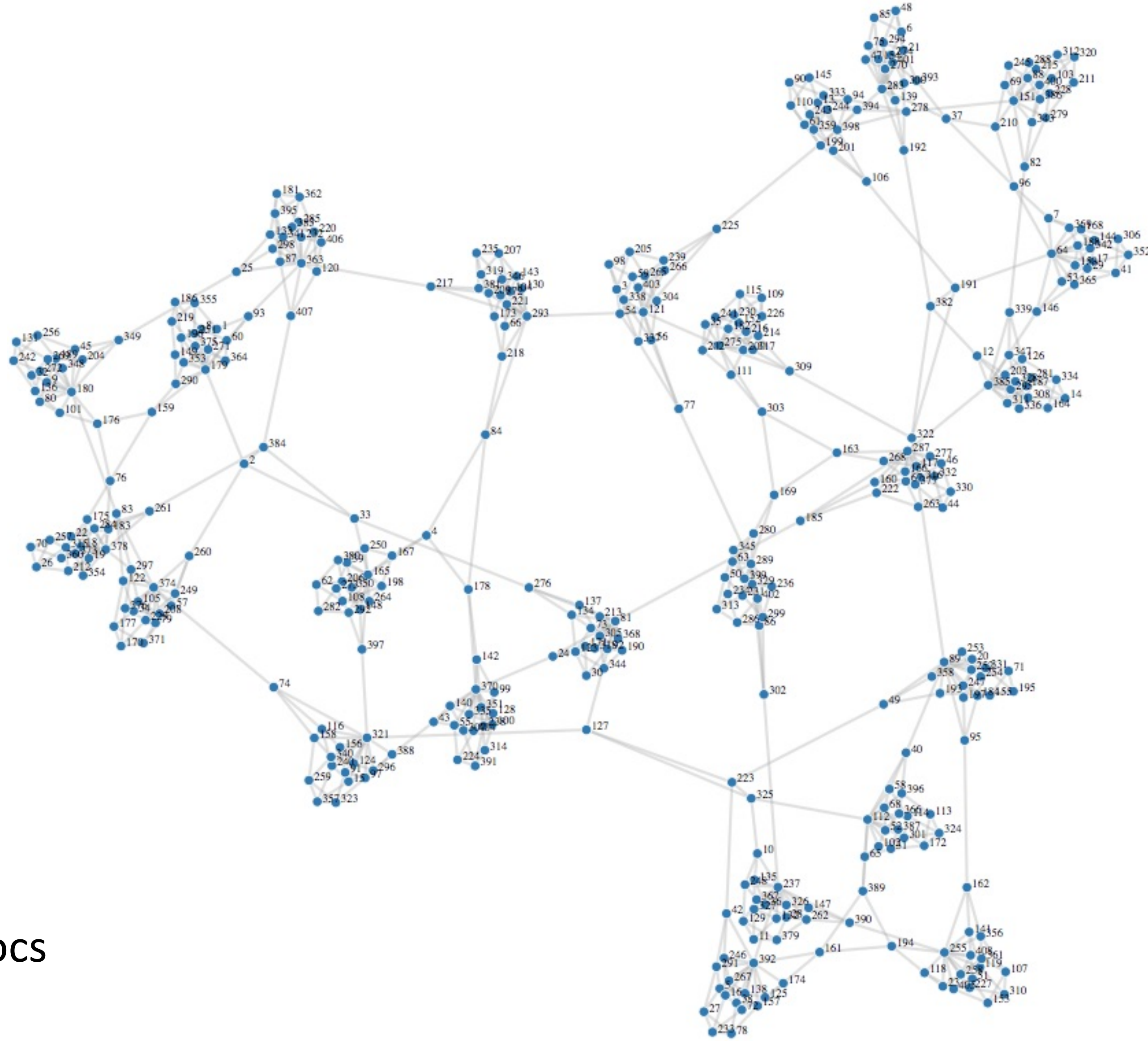
#62: 408 nodes

…

#175: 1014 nodes

…

#876: 5577 nodes



4 edge connected

3 vertex connected

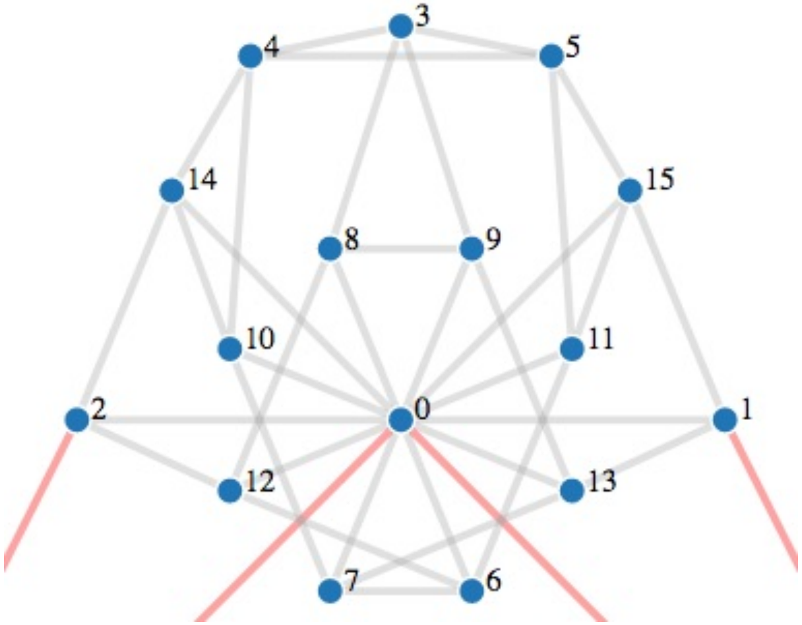
Many identical blocs

Substitution

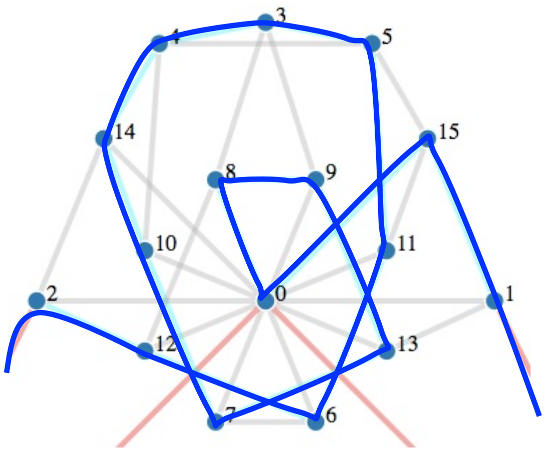
Characteristics

- 16 nodes
- 0 has degree 14

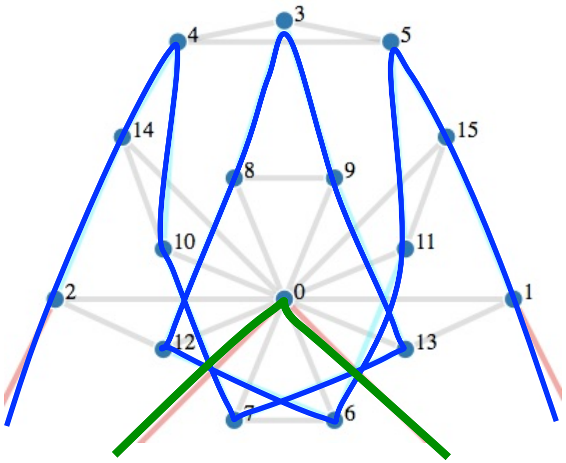
- 0 has 2 edges to the outside
- 1 & 2 have 1 edge to outside



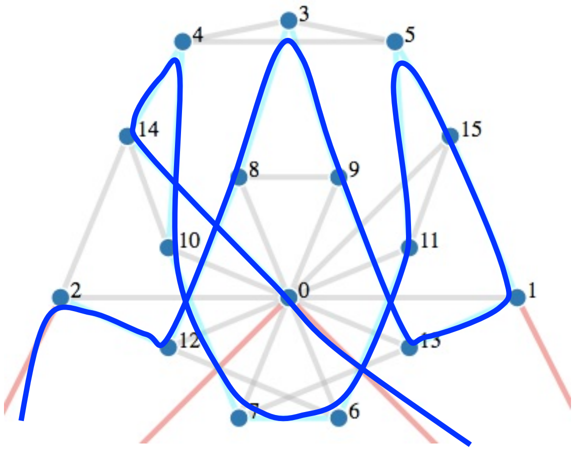
Substitution



2 → visit all nodes → 1

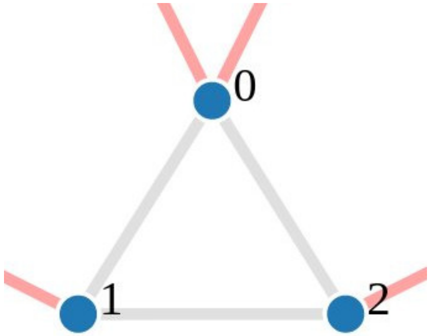


2 → visit all nodes **but 0** → 1



2 → visit all nodes → 0

Equivalent pattern for the outside:



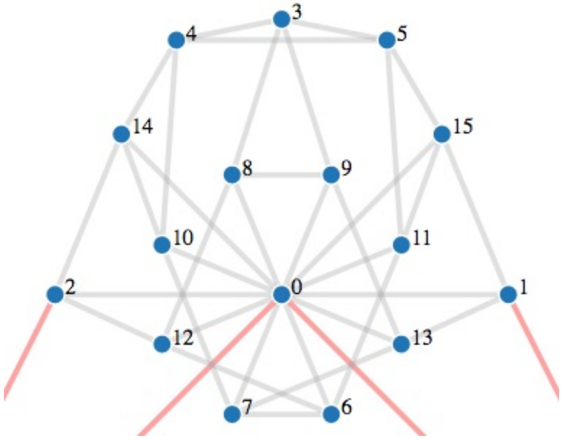
Substitution -- Trick

Subgraph search

- More time consuming that isomorphism

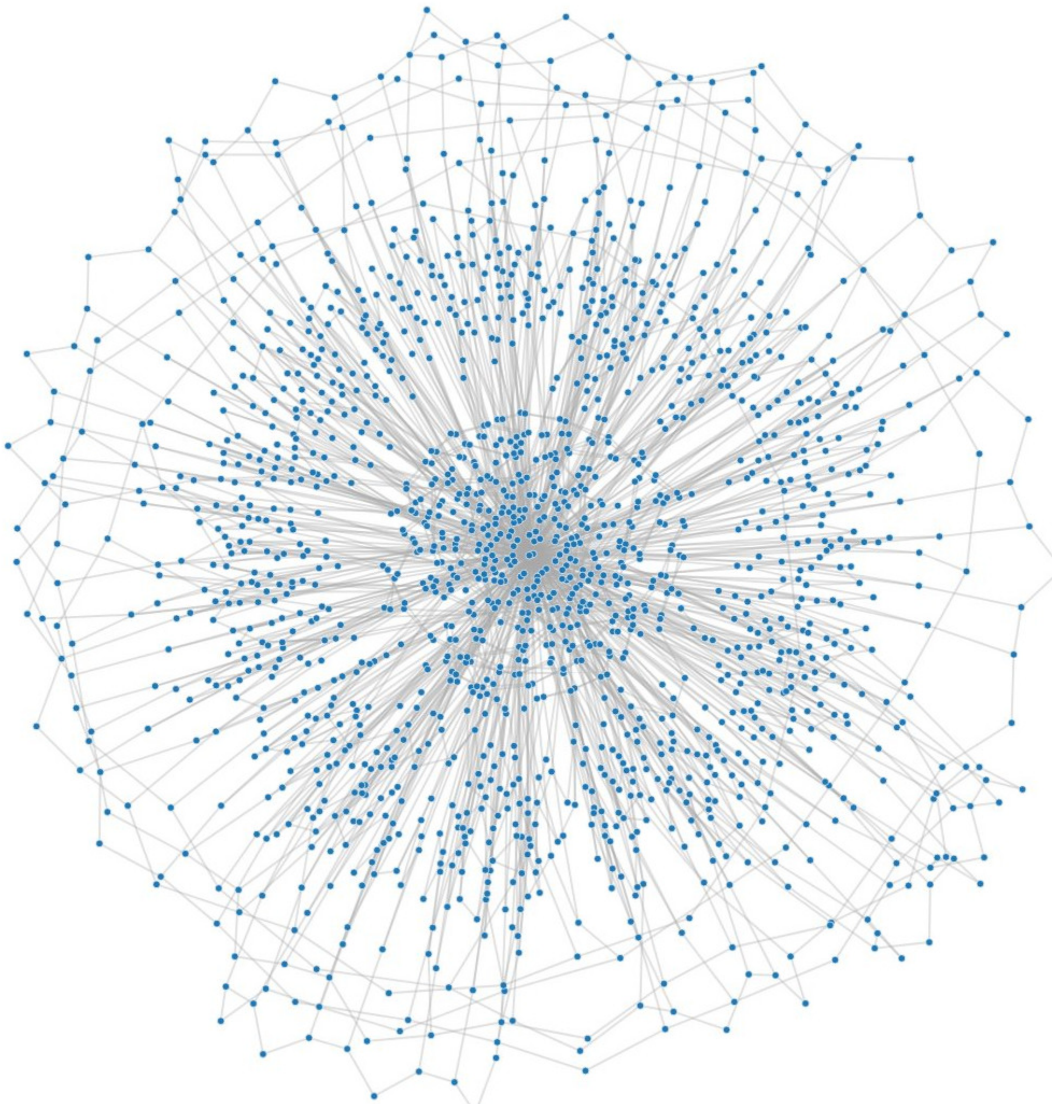
In practice

- Search for a vertex of degree 14
- Identify vertices at distance 2 (BFS)
- Extract subgraph
- Do subgraph search in it
  - Use algorithm from Sagemath



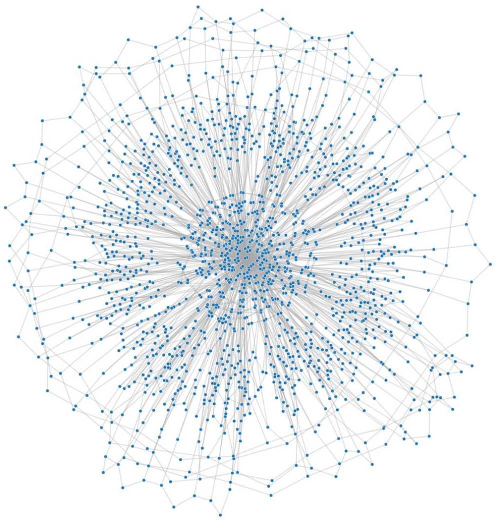
id	n	ILP	n’	ILP	min cut
48	338	>> 12h	78	2 sec	2
62	408	--	96	8 sec	4
…	…	…	…	…	…
875	5576	--	1520	220 sec	4
876	5577	--	1287	118 sec	2

Graphs with 2 nodes of large degree



Graphs with 2 nodes of large degree

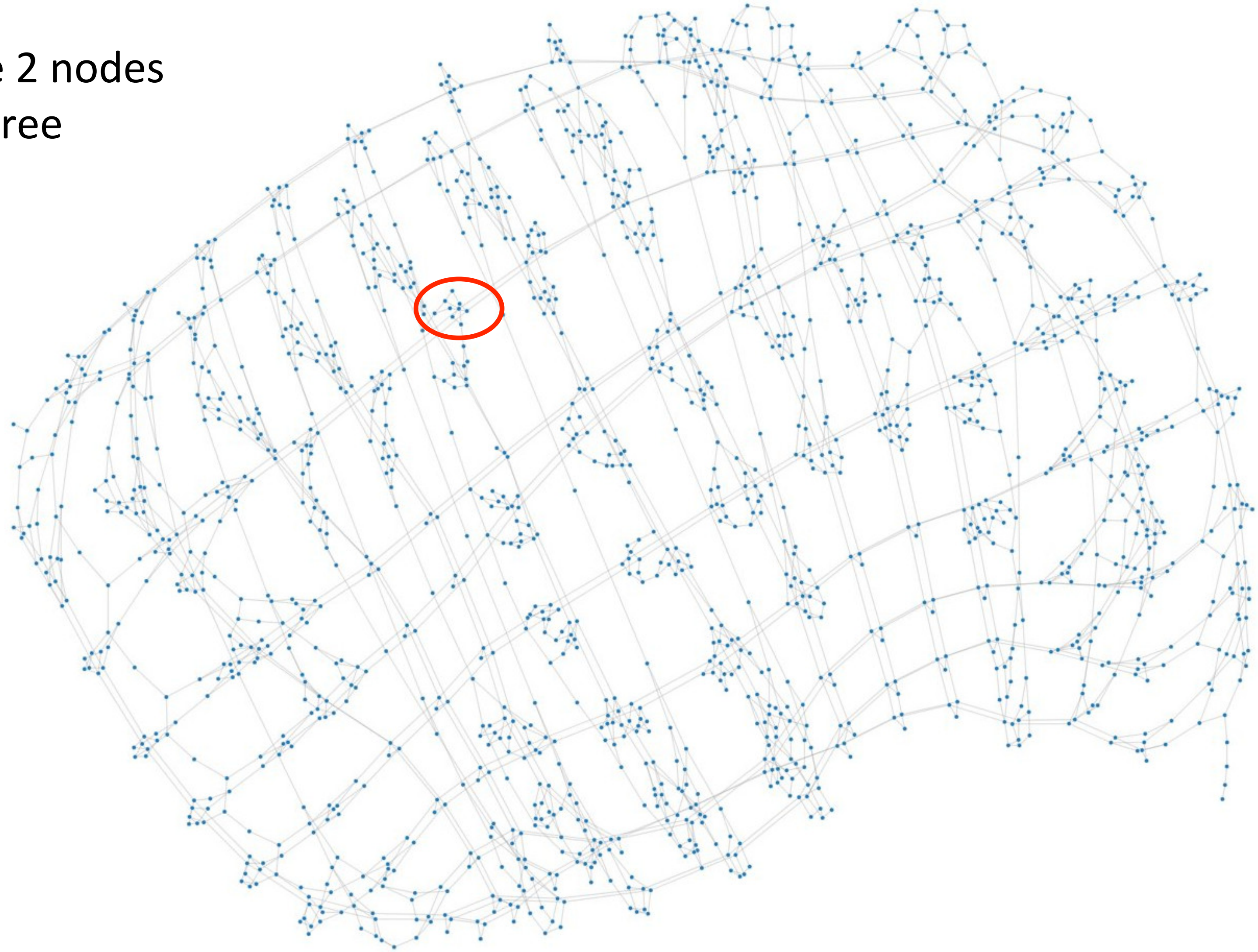
id	n	# deg 2	# small degree	# large degree
268	1644	550	1642 (≤ 6)	2 (192)
846	5244	1749	5242 (≤ 12)	2 (337 & 344)



Reduction rule for degree 2 fails

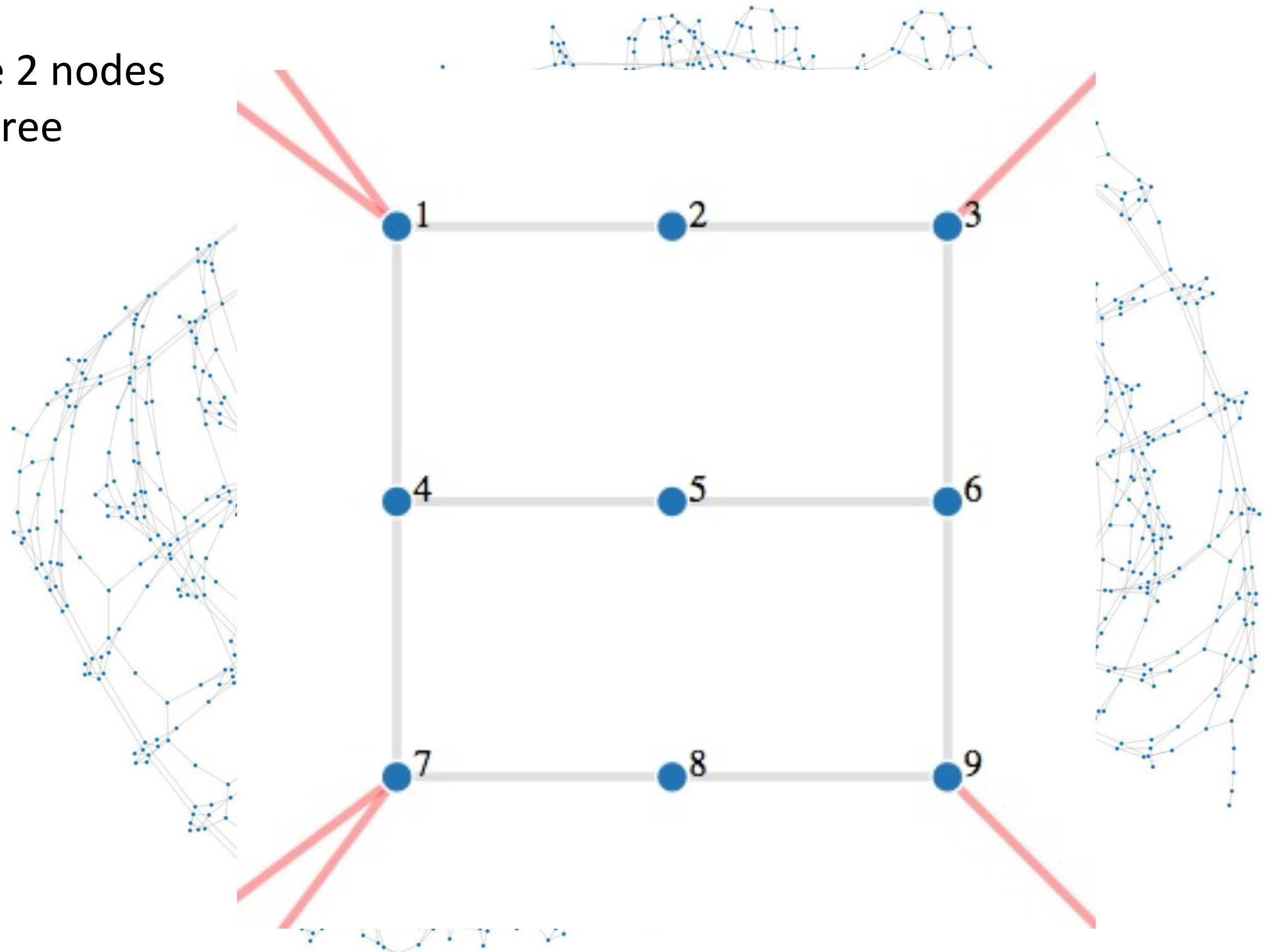
Trying to understand the structure

Remove the 2 nodes of large degree



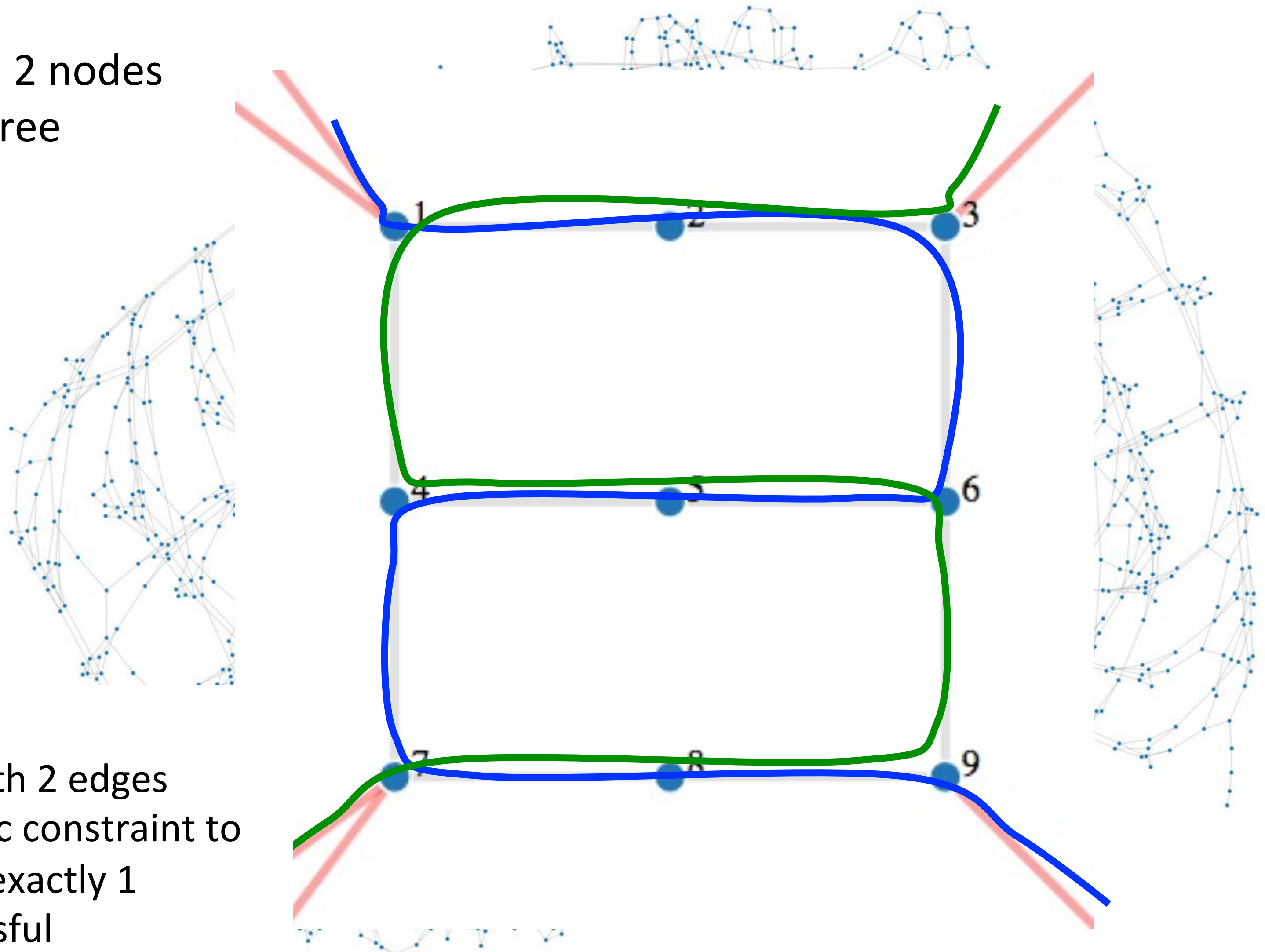
Trying to understand the structure

Remove the 2 nodes of large degree



Trying to understand the structure

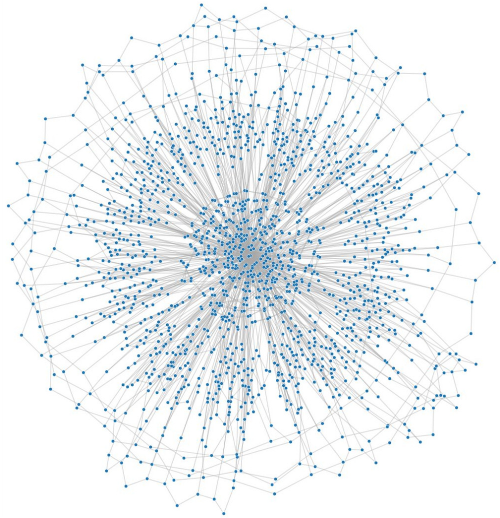
Remove the 2 nodes of large degree



Trial

- Replace with 2 edges
- Add specific constraint to ILP: select exactly 1
- Not successful

Guess & Try



Observation

- The node u with largest degree has exactly 1 neighbor v of degree 2
  - $\longrightarrow$ edge uv in hamiltonian cycle

Specific method

- **Guess** neighbor w of u such that edge uw in hamiltonian cycle
- Remove other incident edges of u
- **Try** to solve hamiltonian cycle with ILP
  - Success $\longrightarrow$ done
  - Infeasible $\longrightarrow$ discard edge and repeat with other neighbor

id	n	Guess & Try	ILP
268	1644	3 sec	6 sec
846	5244	171 sec	536 sec

Guess & Try

General method

- For each vertex incident to only 1 vertex of degree 2
  - **Guess & Try** with limited computation time (e.g., 5 sec)

    Success $\longrightarrow$ done

    Infeasible $\longrightarrow$ discard edge & apply reduction rule for degree 2 (propagation)

    Time limit $\longrightarrow$ continue with next neighbor

- Reveals successful for many graphs without clear/understood structure

Graphs with 5 nodes of large degree

130 graphs

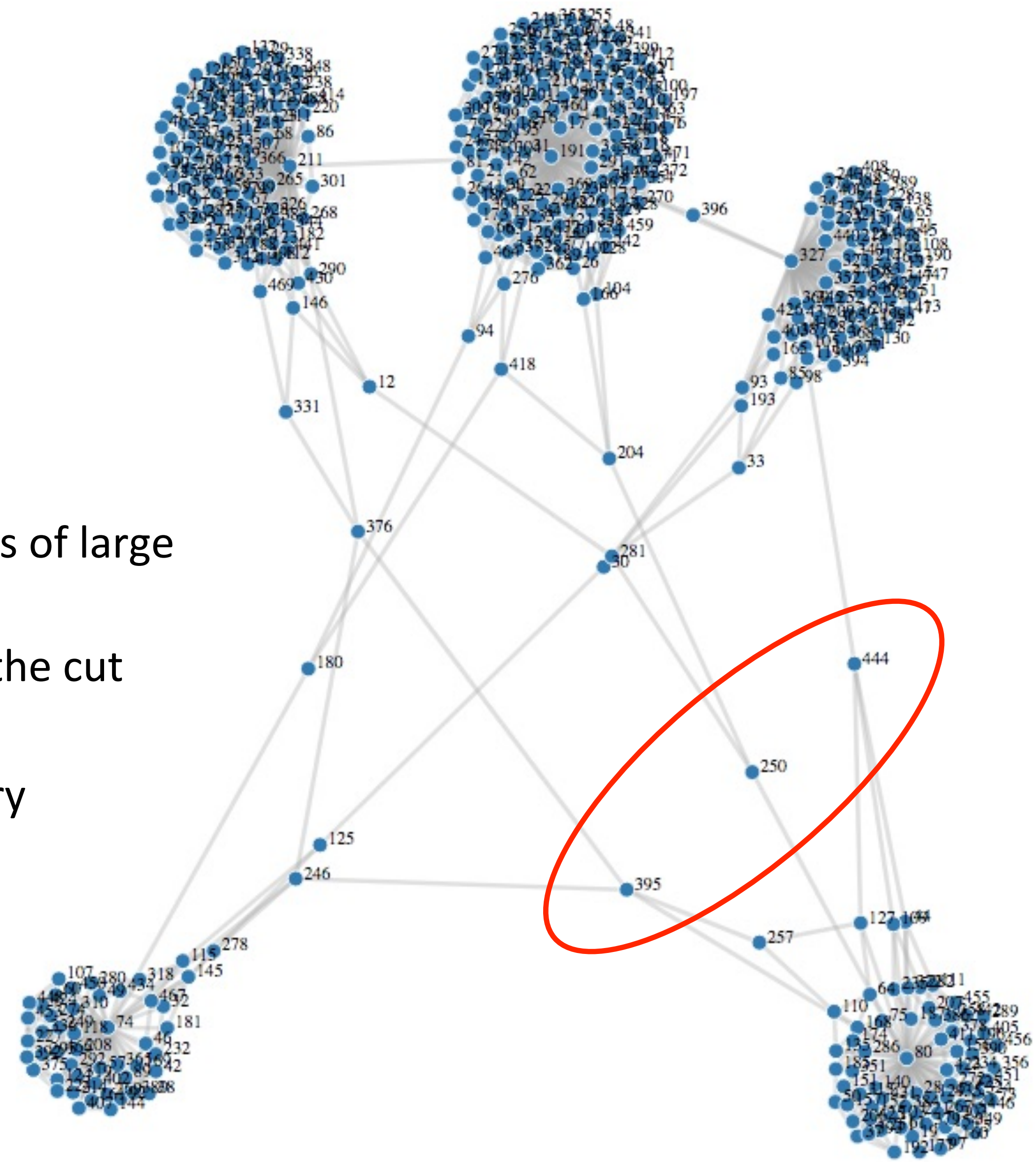
#76 471 nodes

…

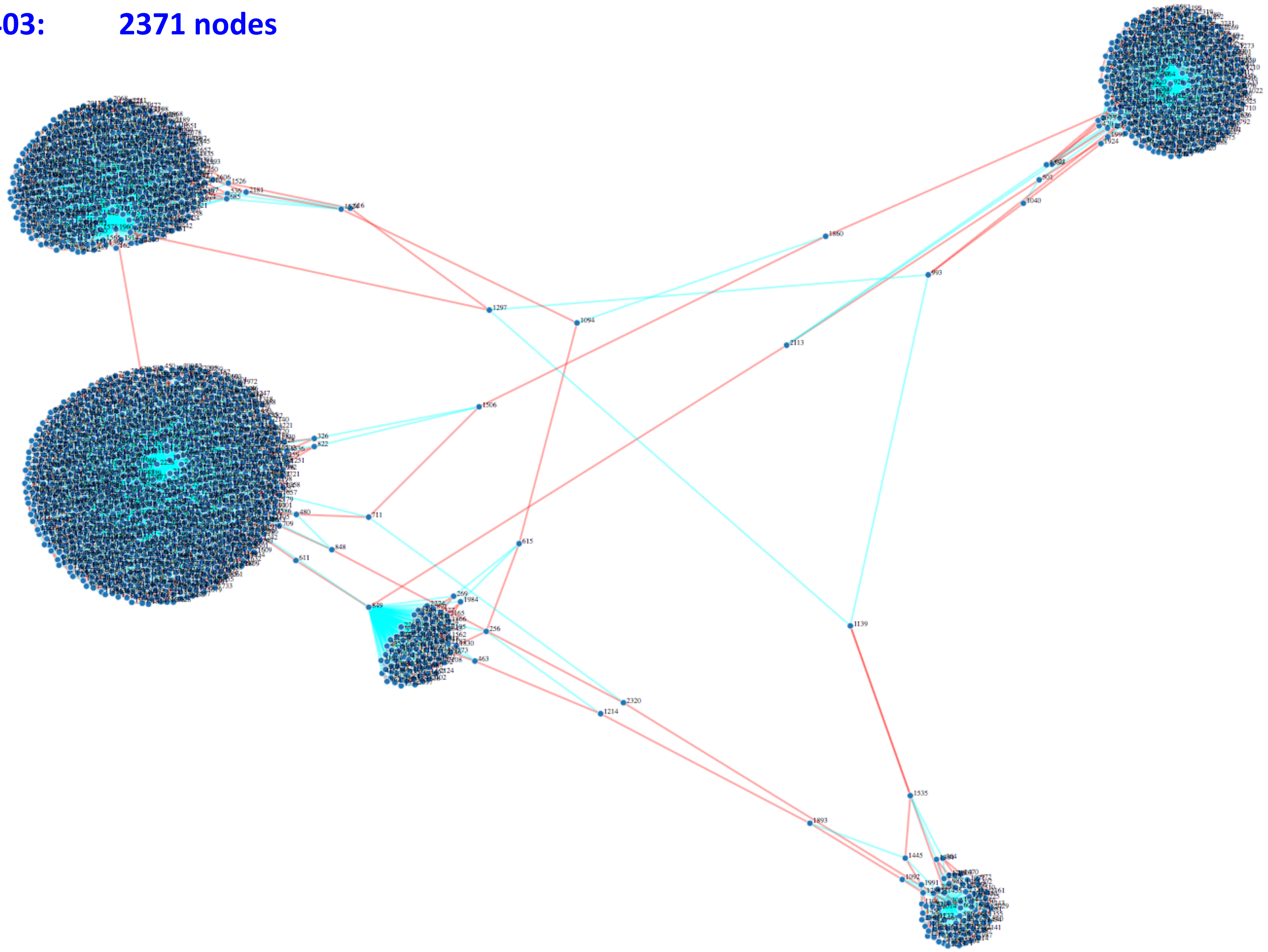
#996 8550 nodes

Idea

- Find small cuts between vertices of large degree (e.g., 3 vertices)
- Try all possibilities for crossing the cut
  - → Similar idea than substitution
- Solve subgraphs with guess & try
- Acceptable search space

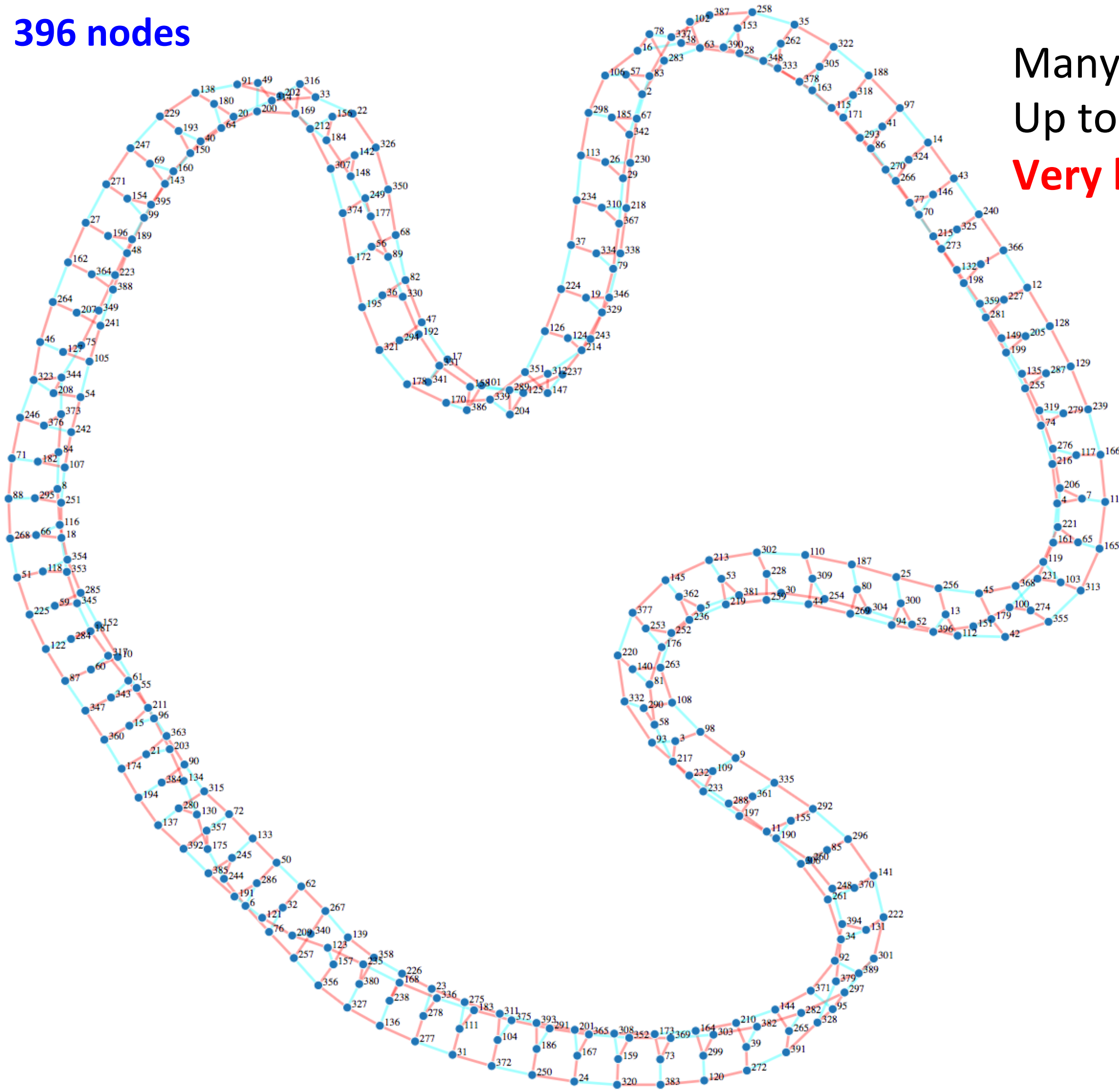


#403: 2371 nodes



Wheels

#58: 396 nodes



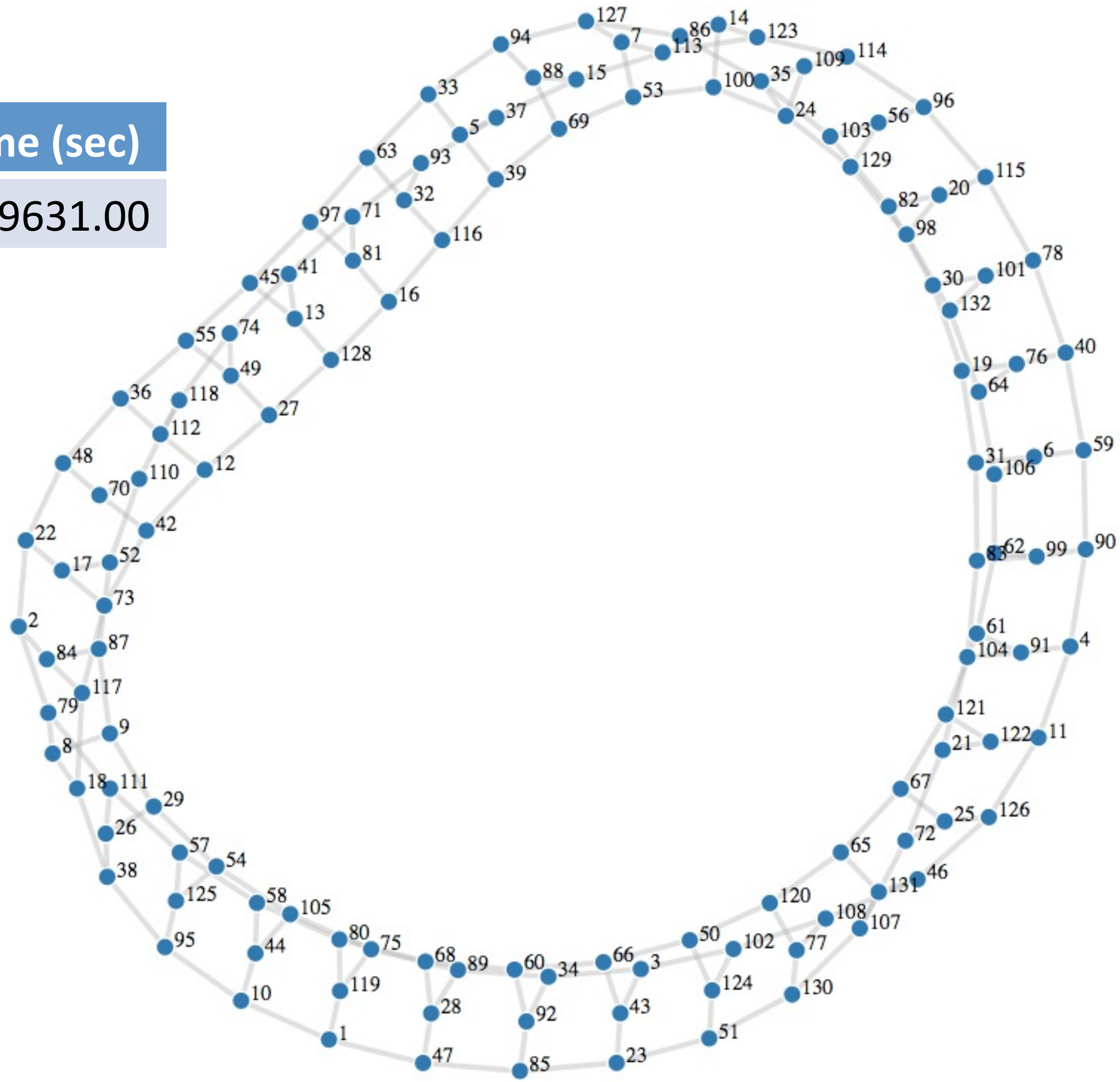
Many graphs with $3 \leq deg \leq 4$

Up to 8886 nodes

Very hard for ILP

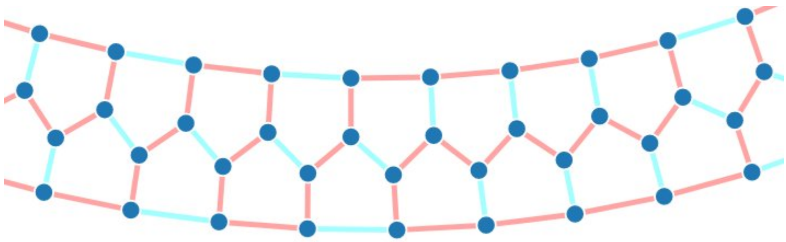
Example #12

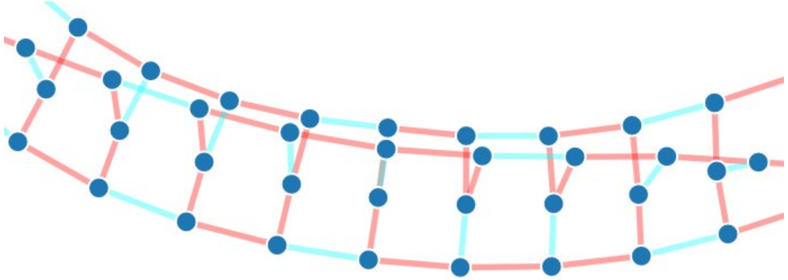
id	n	m	Time (sec)
12	132	199	89631.00



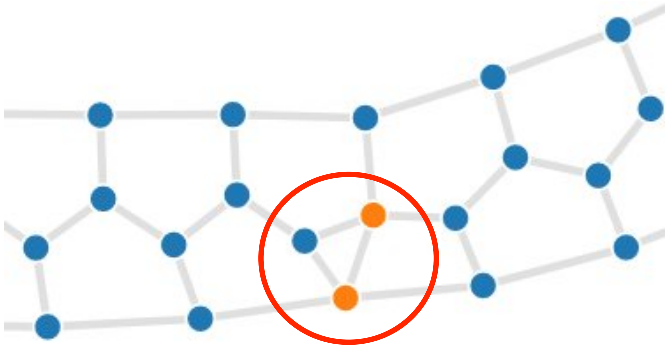
Use visualization tool

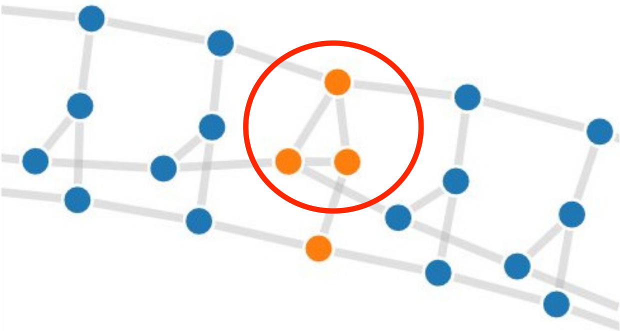
Observe regular pattern …





… with some irregularities …





… and (sometimes) a triangle !

Guess & Try

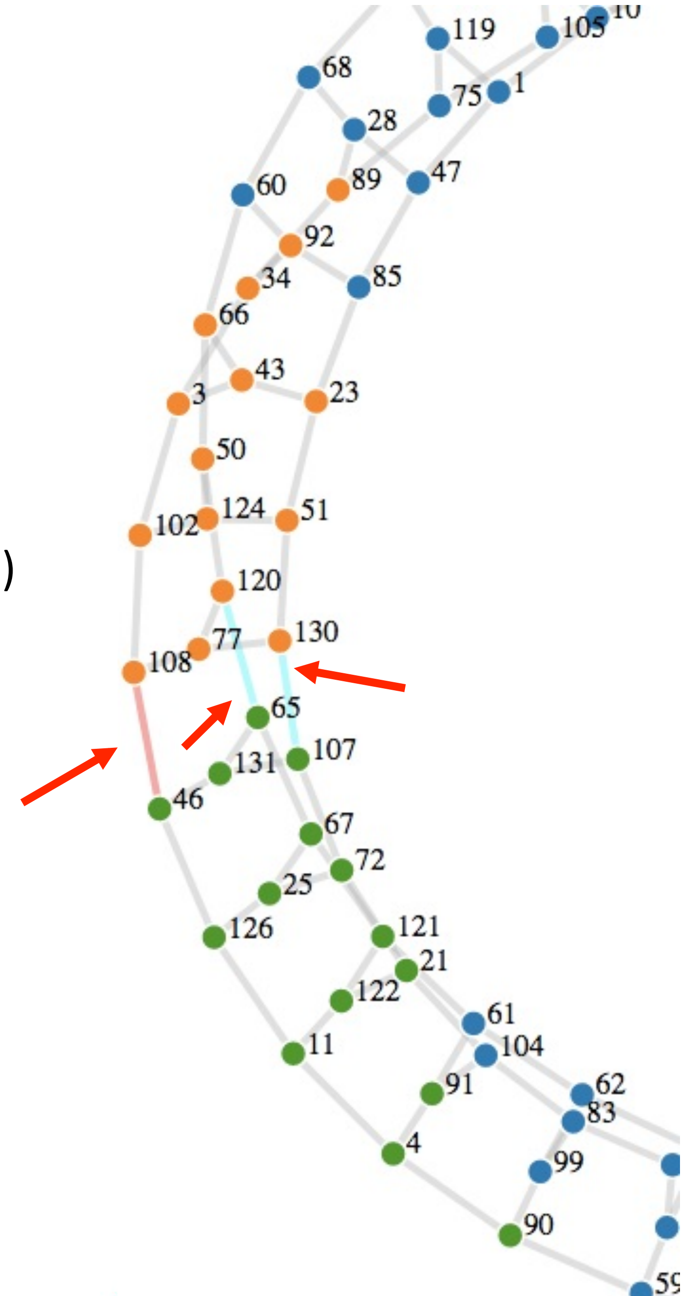
Heuristic

- Select random edge
- Grow disjoint *balls* on each side (distance ≤ 5)

  → Set F of edges incident both balls
- Delete random subset of edges of F (guess)
- **Try** to solve hamiltonian cycle with time limit (30 sec)

  Success → done

  Infeasible or timeout → repeat



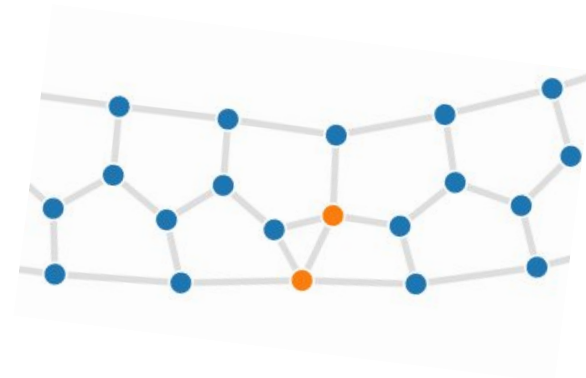
- After several trials, we get the solution

  → Typical overnight runs

Not so nice…

… but successful

Guess & Try



Idea

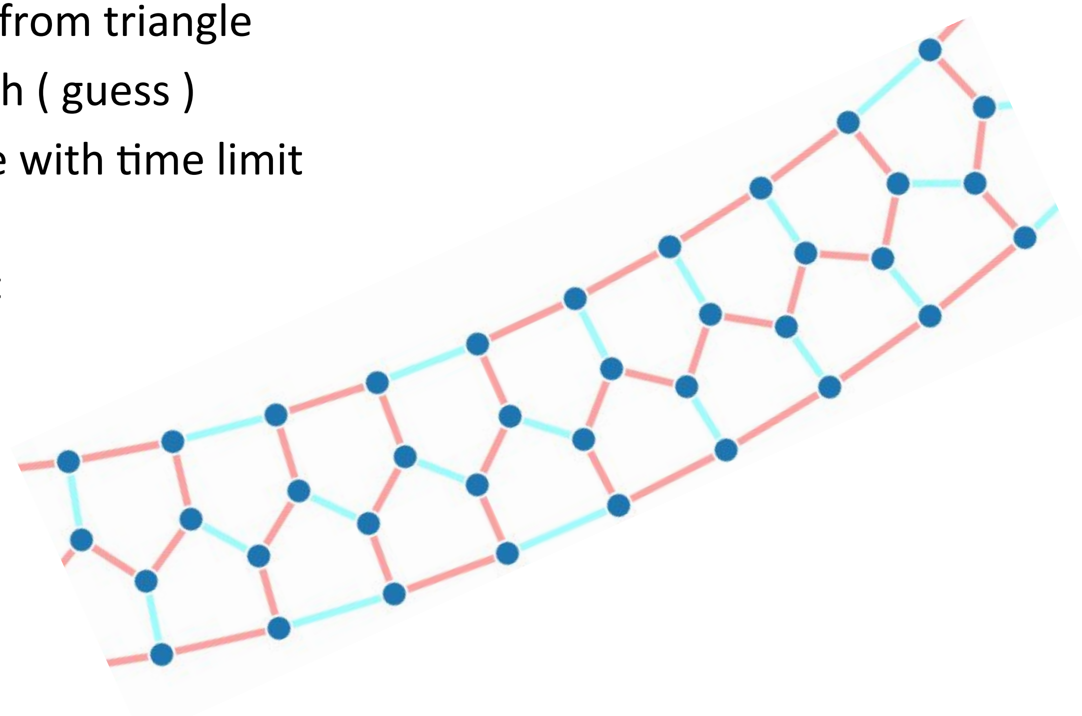
- Let ILP deal with hard part
- Simplify “easy” part by deleting edges

Heuristic

- Find path at the *border* & far from triangle
- Delete some edges of the path (guess)
- **Try** to solve hamiltonian cycle with time limit

  Success → done

  Infeasible or timeout → repeat

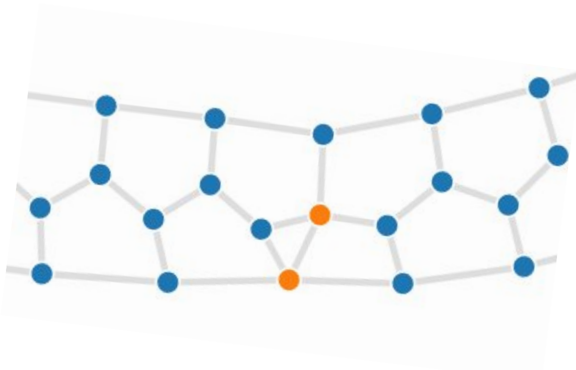


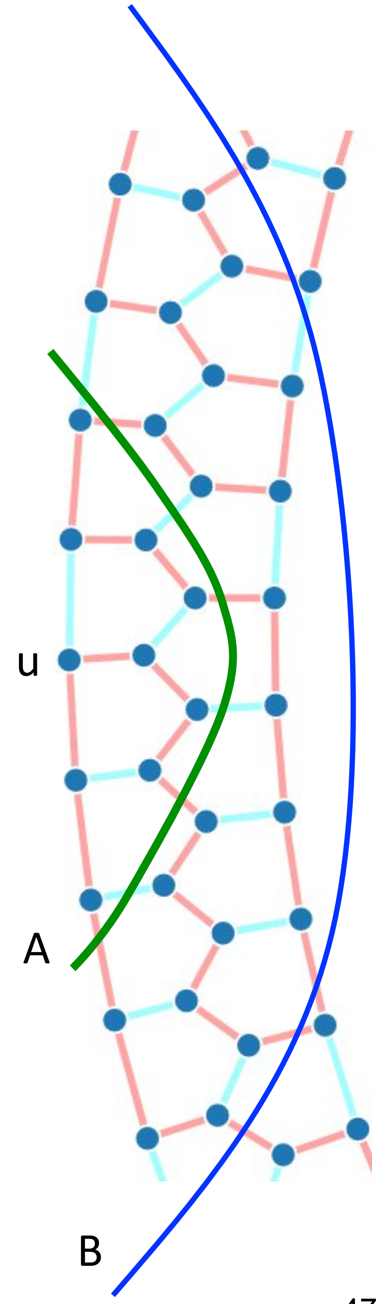
Trick

How to find vertices at the *border* ?

- A = set of vertices at distance ≤ 2 from u (BFS)
- B = set of vertices at distance ≤ 5 from u (BFS)
- Test if   G[B] – A is connected

- Avoid testing vertices close to a triangle

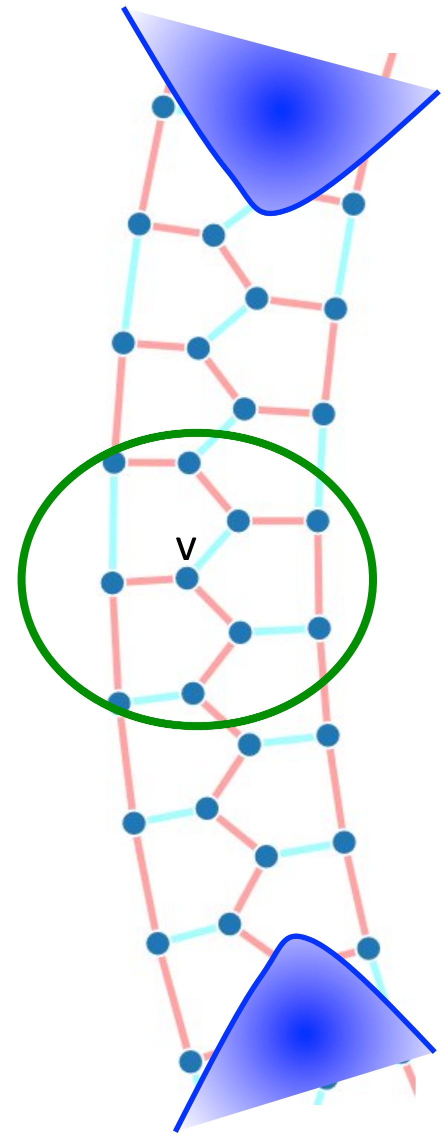




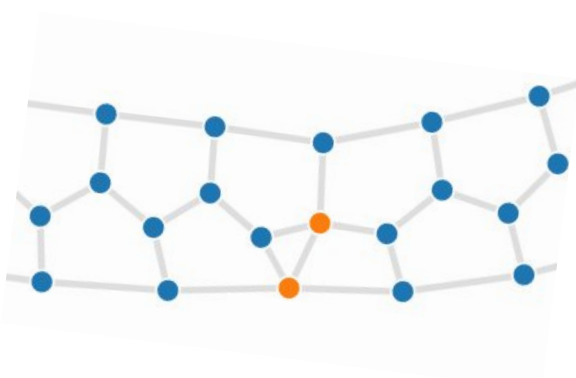
Trick

How to find vertices at the *border* ?

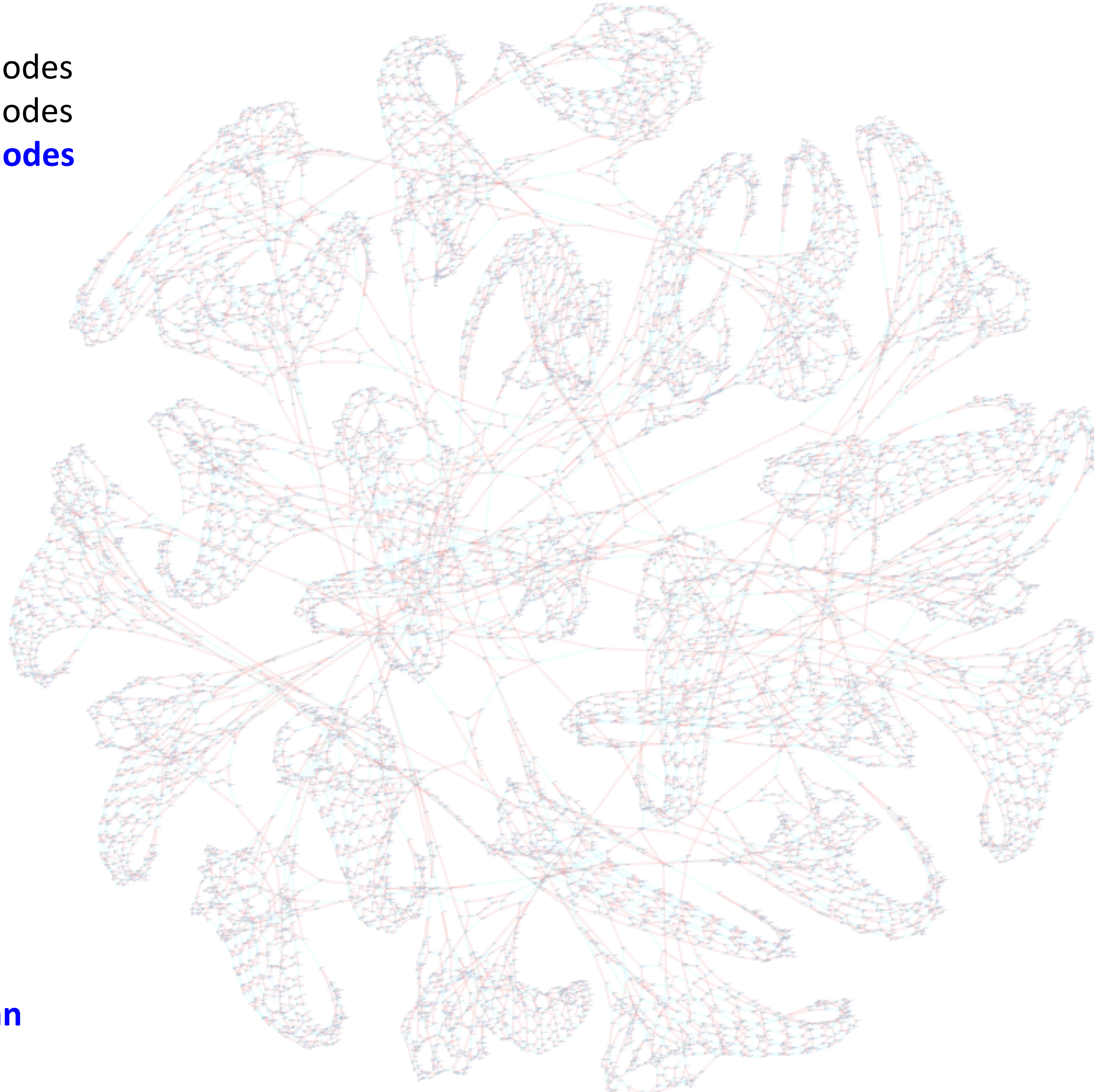
- A = set of vertices at distance ≤ 2 from u (BFS)
- B = set of vertices at distance ≤ 5 from u (BFS)
- Test if G[B] – A is connected



- Avoid testing vertices close to a triangle



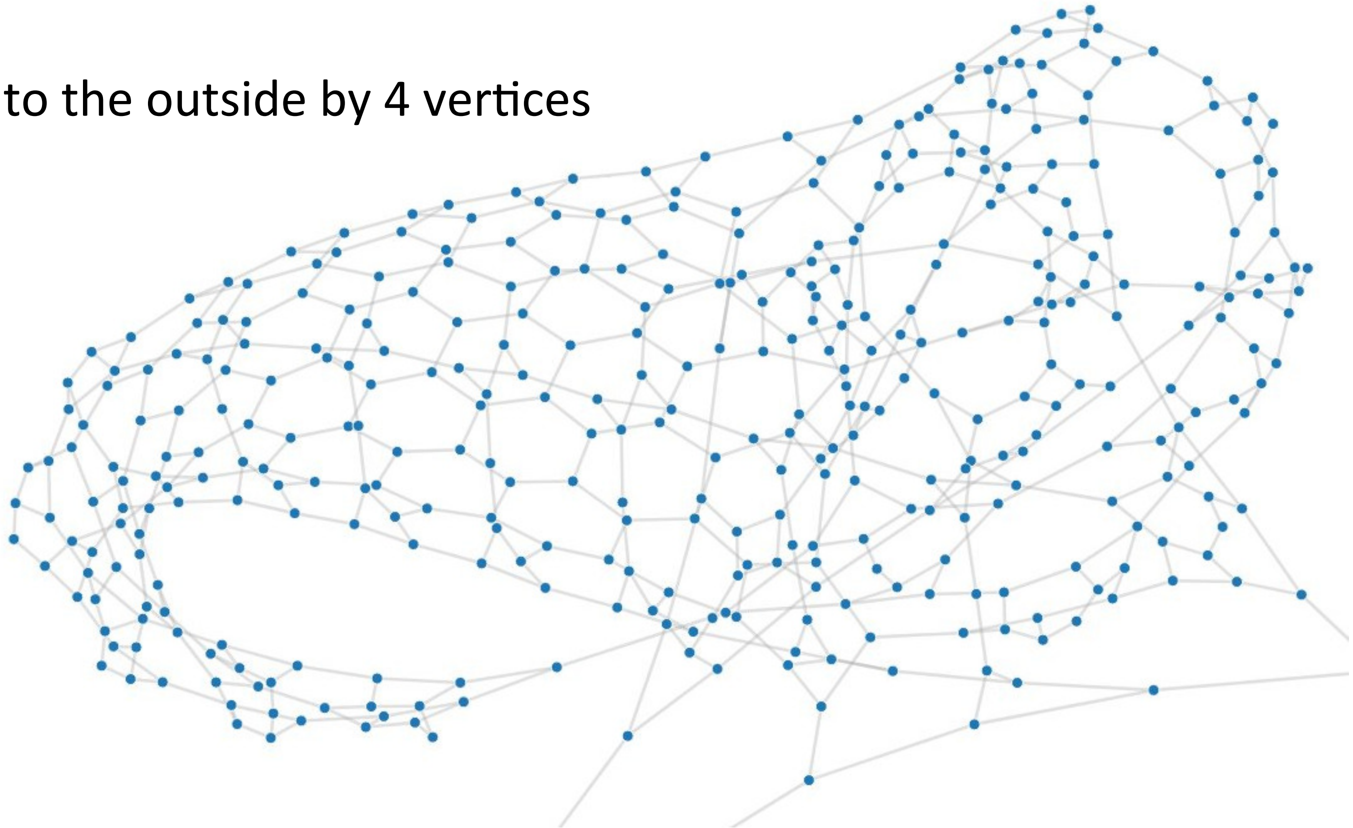
#703	4024 nodes
#989	7918 nodes
#1001	**9528 nodes**



Solved by Nathann in 30 min

Solving 1001

Blocks connected to the outside by 4 vertices



Method

- Compute hamiltonian paths between entry points in the block
- Try all possible substitutions

Outline

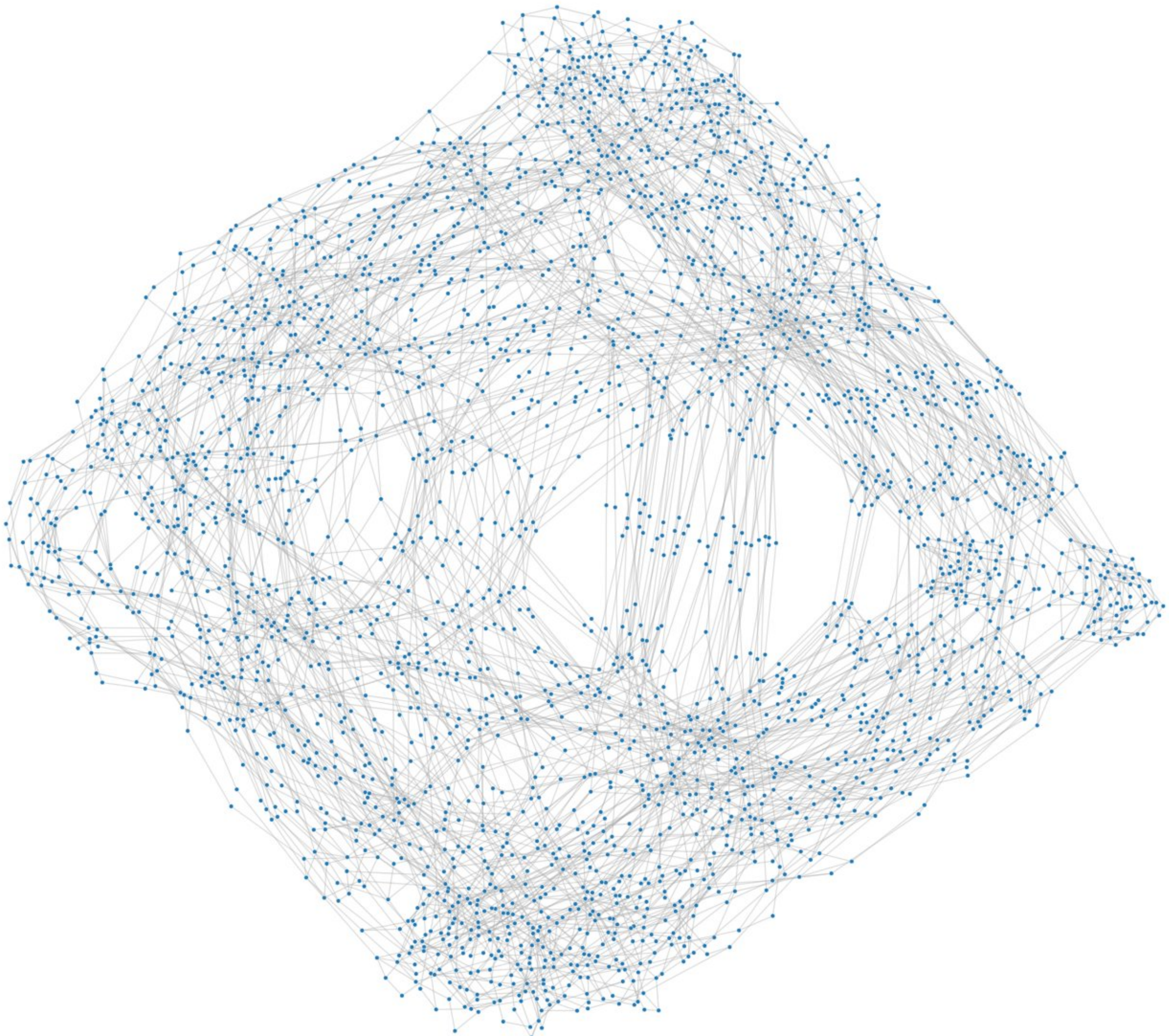
- ✓ ILP formulation
- ✓ Quick exploration of the graphs
- ✓ Resolution methods & tricks
- **Unsolved graphs**

Unsolved graphs

16 (big) graphs

But only 3 blocks

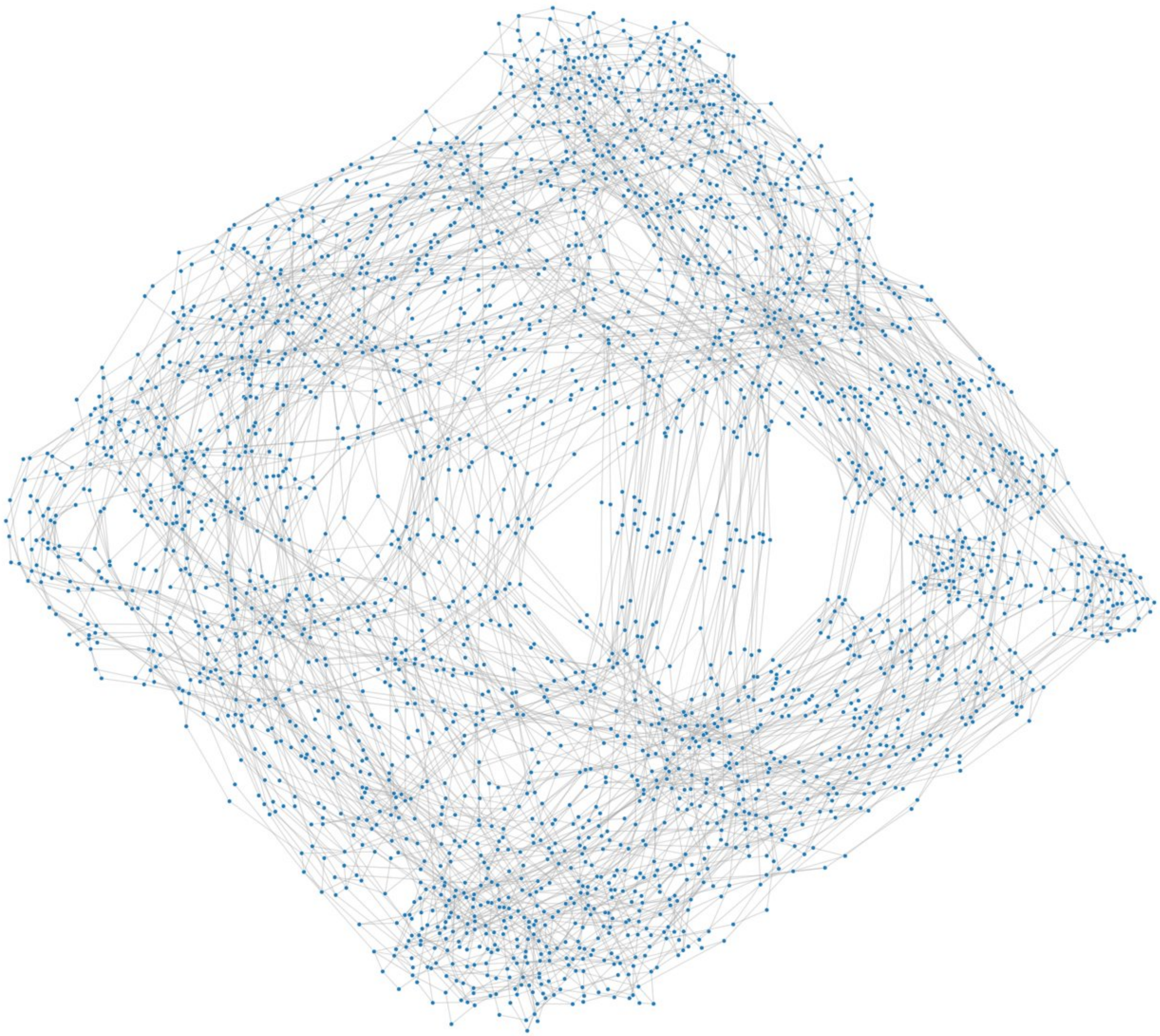
4620, 5544, 6930 nodes



Exploring the structure

Block of order 4620

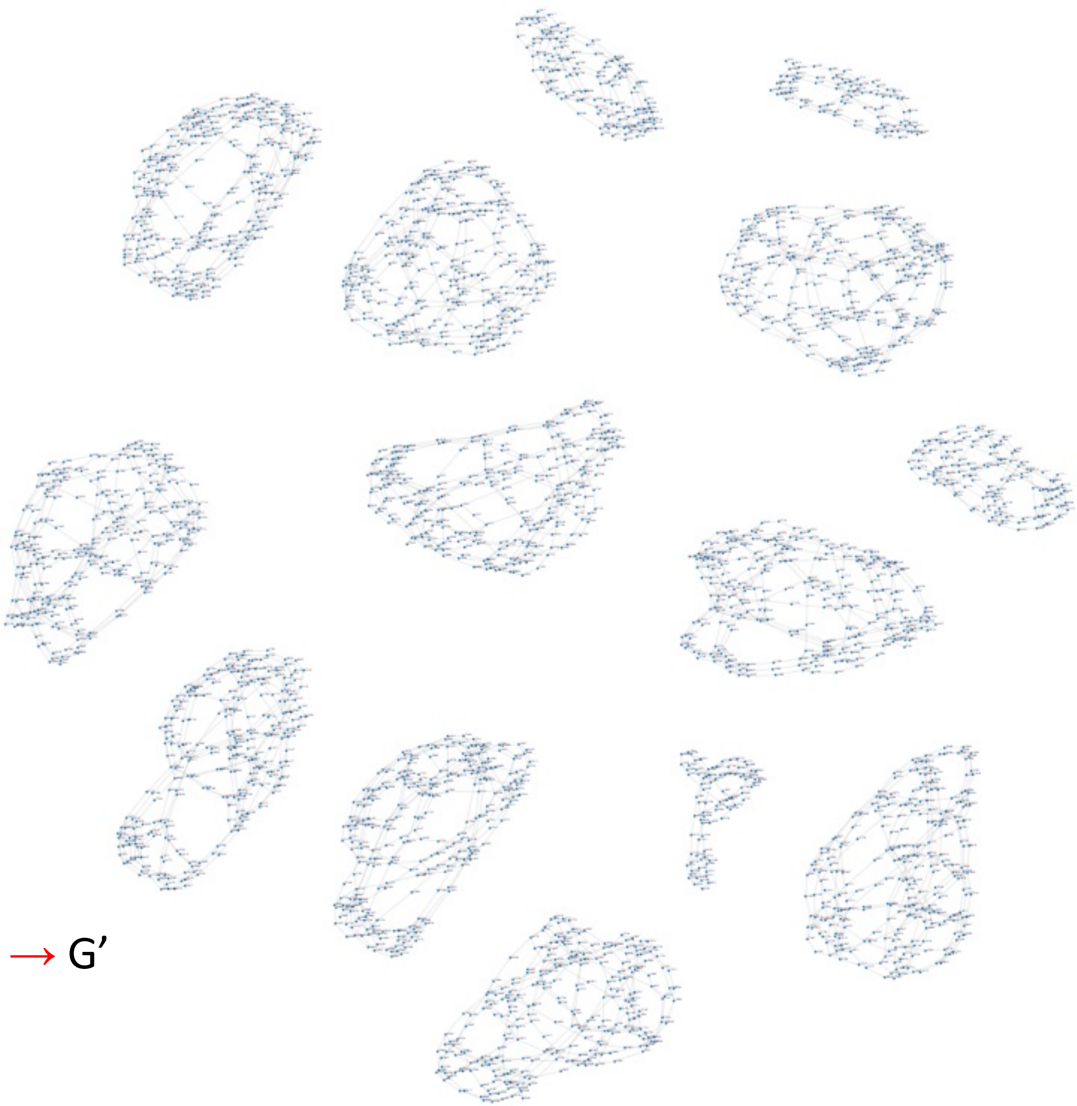
deg	#
2	1540
3	1400
4	560
5	1120



Exploring the structure

Block of order 4620

deg	#
2	1540
3	1400
4	560
5	1120



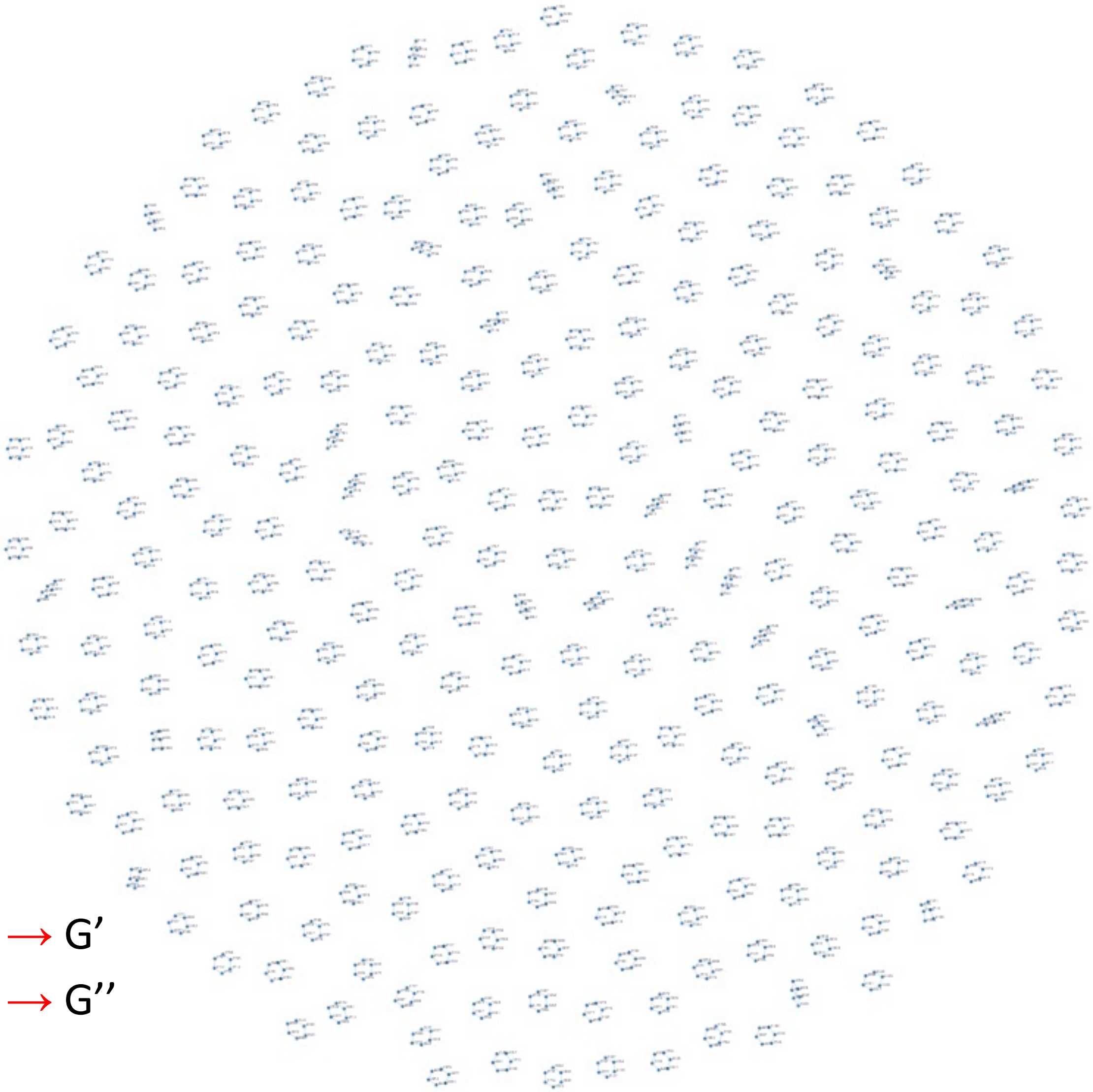
An accidental trial

- Remove vertices of degree 2 $\rightarrow$ G'

Exploring the structure

Block of order 4620

deg	#
2	1540
3	1400
4	560
5	1120



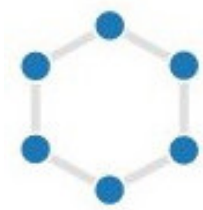
An accidental trial

- Remove vertices of degree 2 $\longrightarrow$ G’
- Remove vertices of degree 2 $\longrightarrow$ G’’

Exploring the structure

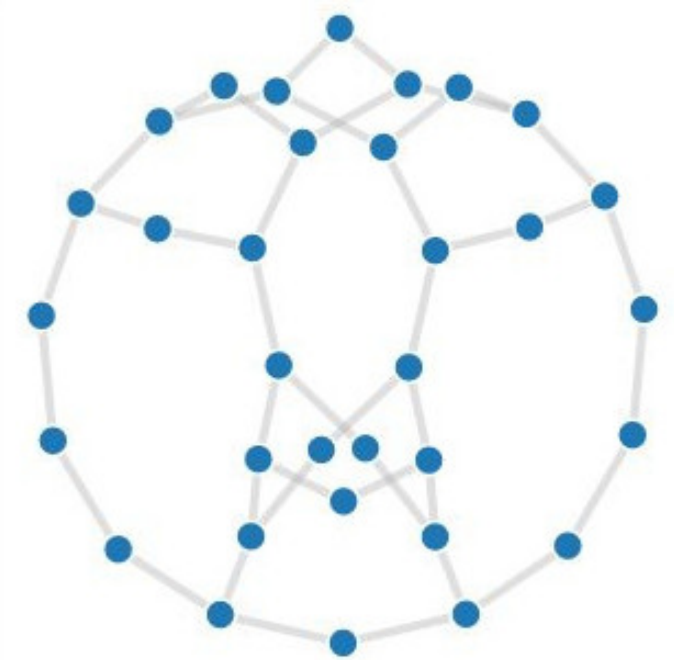
Remove twice vertices of degree 2

- G → G' → G''
- G'' is a collection of disjoint C_6



Removing the edges of the C_6

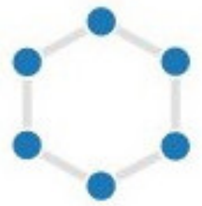
- G – E(G'')
- 140 isomorphic graphs of order 33
  - 140 * 33 = 4620



Exploring the structure

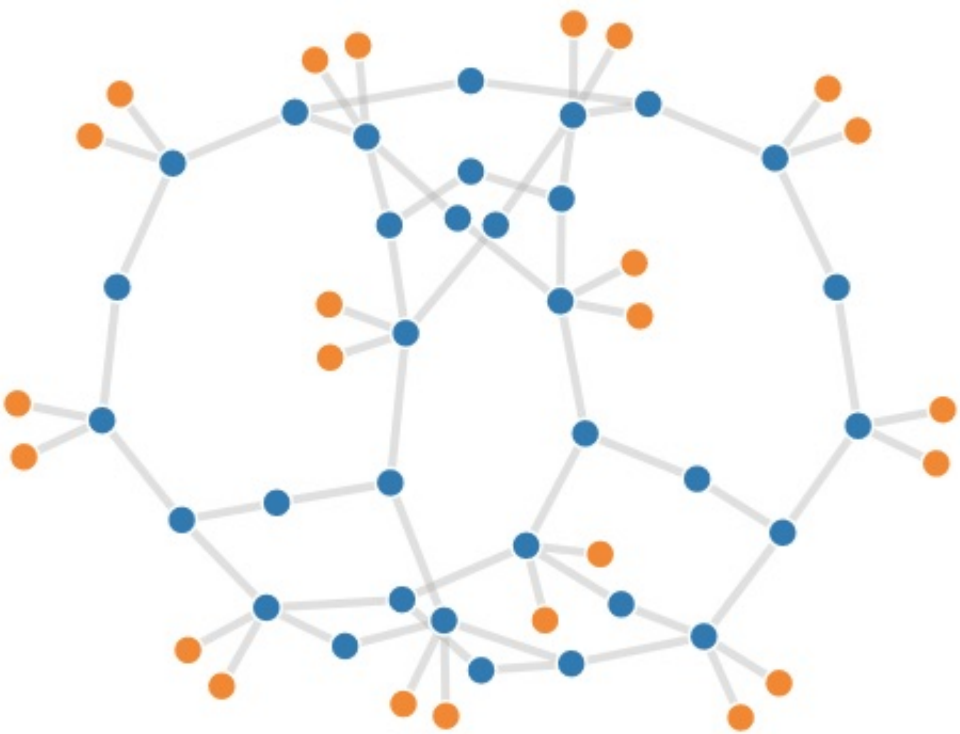
Remove twice vertices of degree 2

- G → G' → G''
- G'' is a collection of disjoint C_6



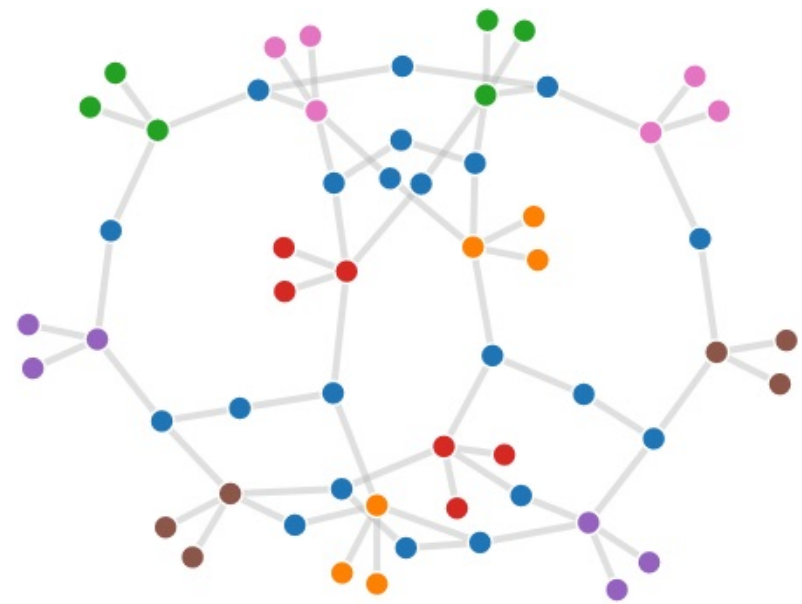
Removing the edges of the C_6

- G – E(G'')
- 140 isomorphic graphs of order 33
  - 140 * 33 = 4620
- Add incident edges



Analyze a block of order 33

- 12 vertices connected to the outside
- **6 hamiltonian paths**, different pairs



Should a solution use a hamiltonian path per block ? YES

How to test ?

- Hamiltonian path : 32 edges
- Try to use at most 31 edges

$$\sum_{v \in N(u)} b(uv) = 2 \qquad \forall u \in V$$

$$\sum_{\text{blocks}} \sum_{uv \in \text{block}} b(uv) \leq 32(\#\text{blocks} - 1) + 31$$

- Infeasible

Not enough

Many many many trials

- Addition of various constraints
- Guess & try hamiltonian paths in blocks to discard edges
- Randomized heuristics
- …

Months of computations

- Code running different rules and exchanging constraints

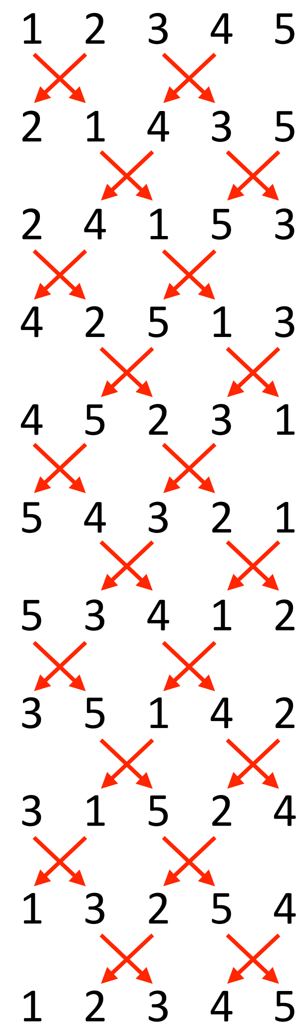
Too hard for us ☹

Bell Ringing -- British campanology

Bell ringing challenge (1600-1650)

- n bells are rung in *rounds*, e.g., order of *pitch* 1, 2, …, n
- Ringers continuously change the order of the bells
  - for as long as possible
  - while not repeating any order
  - and finally return to first round

- **Predicate**: The position of a bell in consecutive rounds change by only 1 (i → i-1, i, i+1)
  - *Permutations in symmetric group on n objects*



Relation to hamiltonian cycle problem

- block → round
- edge → valid move from a round to another

\+ additional transformations

Bell Ringing -- British campanology

Bell ringing challenge (1600-1650)

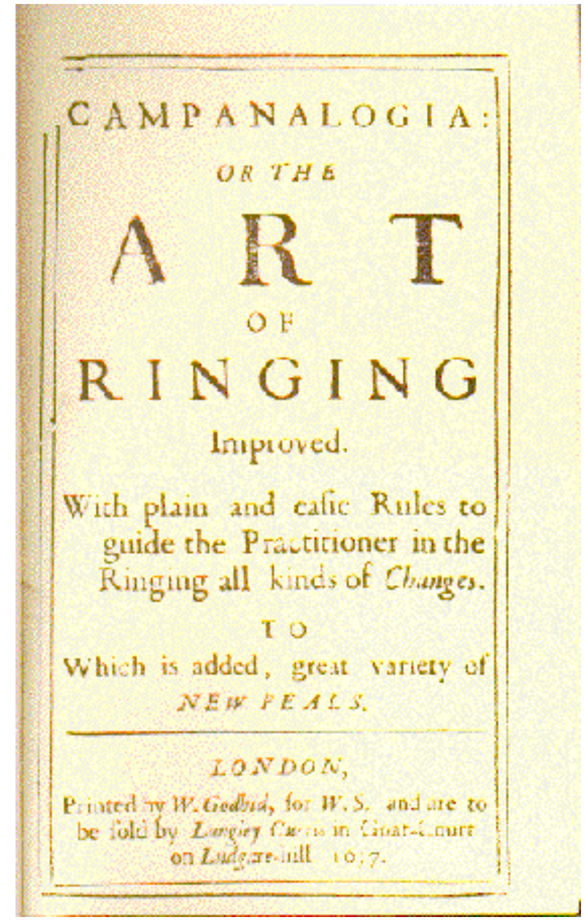
- n bells are rung in *rounds*, e.g., order of *pitch* 1, 2, …, n
- Ringers continuously change the order of the bells
  - for as long as possible
  - while not repeating any order
  - and finally return to first round

- **Predicate**: The position of a bell in consecutive rounds change by only 1 ($i \rightarrow i-1, i, i+1$)
  - *Permutations in symmetric group on n objects*

Relation to hamiltonian cycle problem

- block → round
- edge → valid move from a round to another

\+ additional transformations



CAMPANALOGIA:
OR THE
ART
OF
RINGING
Improved.
With plain and easie Rules to guide the Practitioner in the Ringing all kinds of Changes.
TO
Which is added, great variety of NEW PEALS.

LONDON,
Printed by W. Godbid, for W. S. and are to be sold by Langley Curtis in Goat-Court on Ludgate-hill 1677.

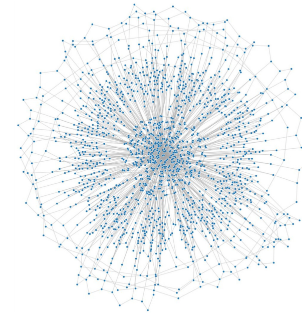
[Fabian Stedman, 1677]

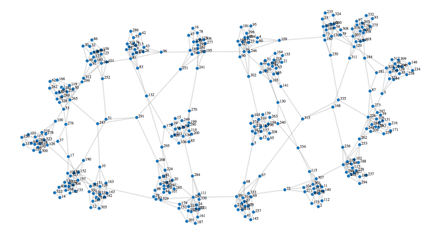
[Stedman: *Tintinnaglia*, 1668]

Summary of methods

Standard methods

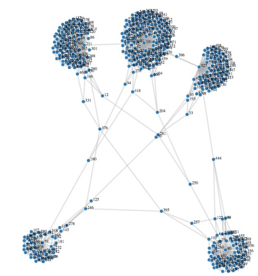
- vertex with 2 neighbors of degree 2 → discard edges
- 2-edge/vertex-cut → split into sub problems
- Guess & try → discard edges



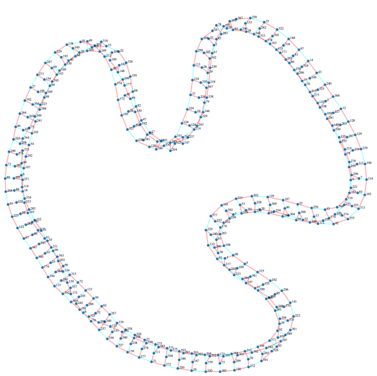


More involved methods

- Substitution with small pattern → reduce size
- 3&4-vertex-cut → split into sub problems



Heuristics



Conclusion

What makes the difference

- **Sagemath**
  - We are **active contributors** of graph module (and others)
  - Very **large collection of graph algorithms**
  - Ease the use of ILP solvers
  - Good visualization tools (**d3.js**)
  - Enable to **quickly test ideas**
- Strong knowledge of **structural graph properties**, **algorithms** & **optimization tools**
- **Manual inspection** to find best strategy for each graph / family
  - Structural properties (classification) & visualization

+ a lot of thinking and craziness

Our contribution to HCP benchmark

- Some instances claimed to be hard can be solved using specific method
- Unsolved instances are really hard (3 graphs)

Take home message

Spend time analyzing input (problem + data)
using all possible means

It's worth the effort

Take notes !

We don't know how we solved some graphs

Thanks

David Coudert

- david.coudert@inria.fr
- http://www-sop.inria.fr/members/David.Coudert

Project-team COATI

- Location: Sophia Antipolis, France
- Combinatorics, Optimization, Algorithms
- http://team.inria.fr/coati