Flinders Hamiltonian Cycle Problem Challenge

Nathann Cohen
CNRS, LRI

David Coudert
COATI
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**
- Hamiltonian by construction
- Designed to be hard to solve using existing algorithms / softwares

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!

Results
- **12 “serious” answers:**
  - **1st** – 985
  - **2nd** – 614
- **avg – 338**
- **FHCP 864**
- **14/12/15**
- **IBM**

He tried

Huge

It’s a race
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
• ...

... 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**
\[
\sum_{v \in N(u)} b(uv) = 2 \quad \forall u \in V \quad (1)
\]

**Constraints for connectivity**
- Spanning tree, flow, cuts, maximum average degree, etc.
ILP formulation -- flow

**Variables**

\[ b(e) = 1 \text{ if edge } e \text{ is selected, 0 otherwise (binary)} \]

\[ f(u,v) \in [0,1] \text{ flow variable per arc (fractional)} \]

**Predicates**

Degree 2 → each vertex incident to 2 selected edges

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

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

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

\[ f(u,v) \leq b(uv) \quad \forall u \in V, \ v \in N(u) \]
ILP formulation -- cuts

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

Predicates:
- Degree 2 → each vertex incident to 2 selected edges
- Connected → spanning tree, flow, cuts, maximum average degree, etc.

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

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

Exponential number of cuts

sum is even
ILP formulation -- cuts

Basic ILP
\[ \forall u \in V, \sum_{v \in N(u)} b(u, v) = 2 \]
solve

Graph of selected edges

connected ?

yes

New constraints
\[ \forall CC, \sum_{uv \in \text{boundary}(CC)} b(u, v) \geq 2 \]

solve

no

Warning: may generate a huge number of constraints → need for reducing instance size
## Running time

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

✓ ILP formulation

• Quick exploration of the graphs

• Resolution methods & tricks

• Unsolved graphs
Use basic graph properties

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>2</td>
<td>→</td>
<td>1908</td>
</tr>
<tr>
<td>Diameter</td>
<td>2</td>
<td>→</td>
<td>1113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Min degree</th>
<th>Max degree</th>
<th>Diameter</th>
<th># graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>399 – 1122</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>124 – 1908</td>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>90 – 392</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>10 – 1113</td>
<td>87</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>11 – 1084</td>
<td>300</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>15 – 111</td>
<td>50</td>
</tr>
</tbody>
</table>
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

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

<table>
<thead>
<tr>
<th>id</th>
<th>$n$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>400</td>
<td>40001</td>
</tr>
<tr>
<td>72</td>
<td>460</td>
<td>52901</td>
</tr>
<tr>
<td>79</td>
<td>480</td>
<td>57601</td>
</tr>
<tr>
<td>84</td>
<td>500</td>
<td>62501</td>
</tr>
<tr>
<td>90</td>
<td>510</td>
<td>65026</td>
</tr>
<tr>
<td>96</td>
<td>540</td>
<td>72901</td>
</tr>
<tr>
<td>128</td>
<td>677</td>
<td>114583</td>
</tr>
<tr>
<td>134</td>
<td>724</td>
<td>131045</td>
</tr>
<tr>
<td>150</td>
<td>823</td>
<td>169333</td>
</tr>
<tr>
<td>162</td>
<td>909</td>
<td>206571</td>
</tr>
<tr>
<td>188</td>
<td>1123</td>
<td>315283</td>
</tr>
</tbody>
</table>

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

- $m = \frac{n^2}{4} + 1$

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

\[ \forall u \in V, \sum_{v \in N(u)} b(u, v) = 2 \]
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 $\gg$ 12h computations

2 edge-cuts + isomorphic blocs

After splitting(s): 2 minutes
# 555

\[ n = 3273 \]

\[ m = 5613 \]
# 555

n = 3273  
m = 5613
Graph isomorphism

A graph $G$ is isomorphic to another graph $H$ if there is an edge-preserving bijection $f: V(G) \rightarrow V(H)$ (i.e., edge $f(u)f(v)$ in $E(H)$ if and only if $uv$ in $E(G)$).

**nauty** [http://pallini.di.uniroma1.it/]

- Very fast
- Return the mapping

[McKay 84; McKay & Piperno 14]
#48: 338 nodes
...
#175: 1014 nodes
...
#875: 5576 nodes

2 edge-cuts
+ isomorphic blocs
# 555

n = 3273

m = 5613
Graph isomorphism -- Trick

Idea: use database of solutions
- Avoid solving twice same instance (from challenge, subgraphs, etc.)

dictionary: graph $\mapsto$ hamiltonian cycle
- Canonical labeling = labeling invariant to isomorphism class
- Immutable graphs are hashable in Sagemath

```python
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: GC = G.canonical_label()
sage: DB = {GC.copy(immutable=True): 123}
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 than 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

<table>
<thead>
<tr>
<th>id</th>
<th>n</th>
<th>ILP</th>
<th>n’</th>
<th>ILP</th>
<th>min cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>338</td>
<td>&gt;&gt; 12h</td>
<td>78</td>
<td>2 sec</td>
<td>2</td>
</tr>
<tr>
<td>62</td>
<td>408</td>
<td>--</td>
<td>96</td>
<td>8 sec</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>875</td>
<td>5576</td>
<td>--</td>
<td>1520</td>
<td>220 sec</td>
<td>4</td>
</tr>
<tr>
<td>876</td>
<td>5577</td>
<td>--</td>
<td>1287</td>
<td>118 sec</td>
<td>2</td>
</tr>
</tbody>
</table>
Graphs with 2 nodes of large degree
Graphs with 2 nodes of large degree

<table>
<thead>
<tr>
<th>id</th>
<th>n</th>
<th># deg 2</th>
<th># small degree</th>
<th># large degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>268</td>
<td>1644</td>
<td>550</td>
<td>1642 (≤ 6)</td>
<td>2 (192)</td>
</tr>
<tr>
<td>846</td>
<td>5244</td>
<td>1749</td>
<td>5242 (≤ 12)</td>
<td>2 (337 &amp; 344)</td>
</tr>
</tbody>
</table>

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
  → 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 → done
  Infeasible → discard edge and repeat with other neighbor

<table>
<thead>
<tr>
<th>id</th>
<th>n</th>
<th>Guess &amp; Try</th>
<th>ILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>268</td>
<td>1644</td>
<td>3 sec</td>
<td>6 sec</td>
</tr>
<tr>
<td>846</td>
<td>5244</td>
<td>171 sec</td>
<td>536 sec</td>
</tr>
</tbody>
</table>
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 → done
    Infeasible → discard edge & apply reduction rule for degree 2 (propagation)
    Time limit → 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 $3 \leq \text{deg} \leq 4$
Up to 8886 nodes
Very hard for ILP
Example #12

<table>
<thead>
<tr>
<th>id</th>
<th>n</th>
<th>m</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>132</td>
<td>199</td>
<td>89631.00</td>
</tr>
</tbody>
</table>
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 $\leq 5$)
  - $\rightarrow$ 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  $\rightarrow$ done
  - Infeasible or timeout  $\rightarrow$ repeat

- After several trials, we get the solution
  $\rightarrow$ 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 $\rightarrow$ done
  Infeasible or timeout $\rightarrow$ 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 $\leq 2$ from $u$ (BFS)
- $B =$ set of vertices at distance $\leq 5$ 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

<table>
<thead>
<tr>
<th>deg</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1540</td>
</tr>
<tr>
<td>3</td>
<td>1400</td>
</tr>
<tr>
<td>4</td>
<td>560</td>
</tr>
<tr>
<td>5</td>
<td>1120</td>
</tr>
</tbody>
</table>
Exploring the structure

Block of order 4620

<table>
<thead>
<tr>
<th>deg</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1540</td>
</tr>
<tr>
<td>3</td>
<td>1400</td>
</tr>
<tr>
<td>4</td>
<td>560</td>
</tr>
<tr>
<td>5</td>
<td>1120</td>
</tr>
</tbody>
</table>

An accidental trial

- Remove vertices of degree 2 $\rightarrow G'$
Exploring the structure

Block of order 4620

<table>
<thead>
<tr>
<th>deg</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1540</td>
</tr>
<tr>
<td>3</td>
<td>1400</td>
</tr>
<tr>
<td>4</td>
<td>560</td>
</tr>
<tr>
<td>5</td>
<td>1120</td>
</tr>
</tbody>
</table>

An accidental trial
- Remove vertices of degree 2 $\rightarrow G'$
- Remove vertices of degree 2 $\rightarrow G''$
Exploring the structure

Remove twice vertices of degree 2
- \( G \rightarrow G' \rightarrow G'' \)
- \( G'' \) is a collection of disjoint \( C_6 \)

Removing the edges of the \( C_6 \)
- \( G - E(G'') \)
- 140 isomorphic graphs of order 33
  - \( 140 \times 33 = 4620 \)
Exploring the structure

Remove twice vertices of degree 2

- $G \rightarrow G' \rightarrow G''$
- $G''$ is a collection of disjoint $C_6$

Removing the edges of the $C_6$

- $G - E(G'')$
- 140 isomorphic graphs of order 33
  - $140 \times 33 = 4620$

- 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
- 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 \( \rightarrow \) round
• edge \( \rightarrow \) valid move from a round to another
  + additional transformations

[Stedman: *Tintinnaglia*, 1668]

[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