An Intelligent Example-Generator for Graph Theory

Michaela Heyer

Boole Centre for Research in Informatics
University College Cork
Ireland
Graph Theory relies heavily on the use of examples to help prove or disprove new conjectures.
Graph Theory relies heavily on the use of examples to help prove or disprove new conjectures - but graphs can be quite large, so examples are hard to find.
Graph Theory relies heavily on the use of examples to help prove or disprove new conjectures - but graphs can be quite large, so examples are hard to find.

Many areas outside Mathematics use graphs.
The Problems

- Graph Theory relies heavily on the use of examples to help prove or disprove new conjectures
  - but graphs can be quite large, so examples are hard to find

- Many areas outside Mathematics use graphs
  - but people in those areas might not have sufficient mathematical knowledge to perform proofs of existence
Create an easy-to-use tool which:
Create an easy-to-use tool which:
- takes in a combination of graph properties
Create an easy-to-use tool which:

- takes in a combination of graph properties
- returns one or more examples of the desired graph
  or outputs that the desired graph has not been found
The Approach

Two main parts:
The Approach

Two main parts:
- generate all graphs up to a given size
The Approach

Two main parts:

- generate all graphs up to a given size
- select those with the required properties
The Challenge: There are lots of graphs!
Generating the Graphs

The Challenge: There are lots of graphs!

<table>
<thead>
<tr>
<th>n</th>
<th>n-node graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>156</td>
</tr>
<tr>
<td>7</td>
<td>1,044</td>
</tr>
<tr>
<td>8</td>
<td>12,346</td>
</tr>
<tr>
<td>9</td>
<td>274,668</td>
</tr>
<tr>
<td>10</td>
<td>12,005,168</td>
</tr>
<tr>
<td>11</td>
<td>1,018,997,864</td>
</tr>
<tr>
<td>12</td>
<td>165,091,172,592</td>
</tr>
</tbody>
</table>
The Challenge: There are lots of graphs!

But luckily...
The Challenge: There are lots of graphs!

But luckily...

- graphs can be generated very quickly
The Challenge: There are lots of graphs!

But luckily...

- graphs can be generated very quickly
- a program for this already exists
  (‘nauty’ by Brendan McKay)
Selecting the Graphs

Should be fast!
Selecting the Graphs

Should be fast!

Suppose that we are looking for a graph with properties $P_1, P_2, \ldots, P_r$
Selecting the Graphs

- Should be fast!
- Suppose that we are looking for a graph with properties $P_1, P_2, \ldots, P_r$
- Idea: Check $P_1, P_2, \ldots, P_r$ simultaneously using parallel processors
Selecting the Graphs

- Should be fast!
- Suppose that we are looking for a graph with properties \( P_1, P_2, \ldots, P_r \)
- Idea: Check \( P_1, P_2, \ldots, P_r \) simultaneously using parallel processors
- For each given graph, let \( t_i \) denote the time to test for property \( P_i \)
Selecting the Graphs

Should be fast!

Suppose that we are looking for a graph with properties $P_1, P_2, \ldots, P_r$

Idea: Check $P_1, P_2, \ldots, P_r$ simultaneously using parallel processors

For each given graph, let $t_i$ denote the time to test for property $P_i$

**TIME:**

- **successful test:** $\max \{ t_i \mid 1 \leq i \leq r \}$
- **unsuccessful test:** $\min \{ t_i \mid 1 \leq i \leq r \ \& \ P_i \ fails \}$
Observation: many graphs will fail rather than pass
Observation: many graphs will fail rather than pass
    ⇒ the quicker we fail, the quicker we can move on
Observation: many graphs will fail rather than pass

⇒ the quicker we fail, the quicker we can move on
⇒ the more properties we have, the higher the chance of a quick fail
Selecting the Graphs cont’d

6 Observation: many graphs will fail rather than pass
   ⇒ the quicker we fail, the quicker we can move on
   ⇒ the more properties we have, the higher the chance of a quick fail

6 Idea: Use an expert system to infer additional properties from the initial list
An “Intelligent” Parallel Solution

From the required properties $P_1, P_2, \ldots, P_r$
infer additional derived properties $P_{r+1}, \ldots, P_s$
An “Intelligent” Parallel Solution

From the required properties $P_1, P_2, \ldots, P_r$
infer additional derived properties $P_{r+1}, \ldots, P_s$

Assume that $s$ processors are available
An “Intelligent” Parallel Solution

- From the required properties $P_1, P_2, \ldots, P_r$
  infer additional derived properties $P_{r+1}, \ldots, P_s$

- Assume that $s$ processors are available

- For each newly-generated graph, test properties $P_1, P_2, \ldots, P_r, P_{r+1}, \ldots, P_s$ in parallel
An “Intelligent” Parallel Solution

6 From the required properties \( P_1, P_2, \ldots, P_r \) infer additional derived properties \( P_{r+1}, \ldots, P_s \)

6 Assume that \( s \) processors are available

6 For each newly-generated graph, test properties \( P_1, P_2, \ldots, P_r, P_{r+1}, \ldots, P_s \) in parallel

6 **TIME:**

- successful test : 
  \[ \max \{ t_j \mid P_j \text{ is the fastest test for } P_i \& 1 \leq i \leq r \} \]

- unsuccessful test : 
  \[ \min \{ t_i \mid 1 \leq i \leq s \& P_i \text{ FAILS} \} \]
The Expert System

Apply theorems to the initial properties $P_1, \ldots, P_r$
to infer additional properties $P_{r+1}, \ldots, P_s$
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$.

Suppose we are looking for a graph with property A.
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$.

Suppose we are looking for a graph with property $A$:

$A \Rightarrow B$:
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$.

Suppose we are looking for a graph with property $A$

$A \Rightarrow B$ : Add $\neg B$ as a fail property
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$.

Suppose we are looking for a graph with property $A$

$A \Rightarrow B$ : Add $\neg B$ as a fail property

$B \Rightarrow A$ :
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$

Suppose we are looking for a graph with property $A$

- $A \Rightarrow B$ : Add $\neg B$ as a fail property
- $B \Rightarrow A$ : Add $B$ as a pass property for $A$
Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$.

Suppose we are looking for a graph with property A,

- $A \Rightarrow B$ : Add $\neg B$ as a fail property
- $B \Rightarrow A$ : Add B as a pass property for A
- $A \Leftrightarrow B$ :
The Expert System

Apply theorems to the initial properties $P_1, \ldots, P_r$ to infer additional properties $P_{r+1}, \ldots, P_s$

Suppose we are looking for a graph with property $A$

$A \Rightarrow B$ : Add $\neg B$ as a fail property

$B \Rightarrow A$ : Add $B$ as a pass property for $A$

$A \Leftrightarrow B$ : Add $B$ as a pass property for $A$ and $\neg B$ as a fail property
Overview of the System

List of Theorems → Query: Required Properties

→ Expert System

→ Graph-Generating S/W

→ Properties to Test

→ Generated Graphs

Cluster:
Run Property-Testing Algorithms in Parallel

→ Output:
Graph with the Required Props OR Fail
Example of Intelligent Solution

Query:
We are looking for a graph that is critical and Hamiltonian
Example of Intelligent Solution

- Query:
  We are looking for a graph that is critical and Hamiltonian

- Required Properties:
  \[ P_1 : G \text{ is critical} \]
  \[ P_2 : G \text{ is Hamiltonian} \]
Infer additional properties...

Some of the Derived Properties in Wish list:
Some of the Derived Properties in Wish list:

\[ P_3 : \alpha < \text{connectivity} \]
\[ P_4 : G \text{ is a hyper cube} \]
\[ P_5 : \delta \geq v/2 \]
Infer additional properties...

Some of the Derived Properties in Wish list:

- $P_3 : \alpha < \text{connectivity}$
- $P_4 : G$ is a hyper cube
- $P_5 : \delta \geq \nu/2$

Some of the Derived Properties in Disprove list:

- $P_6 : G$ is not a block
- $P_7 : G$ is not connected
- $P_8 : G$ is not biconnected
- $P_9 : G$ is bipartite & has bipartition$(XY), |X| \neq |Y|$
Check Graphs...
Check Graphs...
Check Graphs...

- P1 is critical
- $\alpha < \text{connectivity}$
- P13 is hyper cube
- P14 is not a block
- is not connected
- P21 is not biconnected
Check Graphs...

- P1 is critical
- \( \alpha < \text{connectivity} \)
- P13 is hyper cube
- P14 is not a block → True
- is not connected
- P21 is not biconnected
Check Graphs...

- P1 is critical
- $\alpha < \text{connectivity}$
- P13 is hyper cube
- P14 is not a block
- P21 is not biconnected
- True → Not critical

Discard graph!
Check Graphs...
Check Graphs...
Check Graphs...

P1 is critical

alpha < connectivity

P13 is hyper cube

P14 is not a block

P21 is not biconnected
Check Graphs...

- P1 is critical
- $\alpha < \text{connectivity}$
- P13 is hyper cube
- P14 is not a block
- is not connected $\rightarrow$ True
- P21 is not biconnected
Check Graphs...

- P1 is critical
- \( \alpha < \text{connectivity} \)
- P13 is hyper cube
- P14 is not a block
- P21 is not biconnected

True → Not critical
Discard graph!
Check Graphs...
Check Graphs...
Check Graphs...

P1 is critical

alpha < connectivity

P13 is hyper cube

P14 is not a block

is not connected

P21 is not biconnected
Check Graphs...

P1 is critical

alpha < connectivity

P13 is hyper cube

P14 is not a block

is not connected

P21 is not biconnected

True
Check Graphs...

P1 is critical
alpha < connectivity
P13 is hyper cube
P14 is not a block
is not connected
P21 is not biconnected
True → Not Hamiltonian
Discard graph!
Check Graphs...
Check Graphs...
P1 is critical

alpha < connectivity

P13 is hyper cube

P14 is not a block

is not connected

P21 is not biconnected
Check Graphs...

- P1 is critical → True...Wait...
- \( \alpha < \text{connectivity} \)
- P13 is hyper cube
- P14 is not a block
- is not connected
- P21 is not biconnected
Check Graphs...

- P1 is critical → True...Wait...
- alpha < connectivity → True
- P13 is hyper cube →
- P14 is not a block →
- is not connected →
- P21 is not biconnected →
Check Graphs...

P1 is critical → True...Wait...
alpha < connectivity → True

Critical and Hamiltonian:
Keep graph!
P13 is hyper cube
P14 is not a block
is not connected
P21 is not biconnected
The Current System

Expert system written in Haskell
The Current System

- Expert system written in Haskell
- Cluster code written in C++ using MPI
The Current System

- Expert system written in Haskell
- Cluster code written in C++ using MPI
- Runs on the BCRI Compute Cluster
The Current System

- Expert system written in Haskell
- Cluster code written in C++ using MPI
- Runs on the BCRI Compute Cluster
- Collection of approx. 200 theorems
The Current System

- Expert system written in Haskell
- Cluster code written in C++ using MPI
- Runs on the BCRI Compute Cluster
- Collection of approx. 200 theorems
- Approx. 180 algorithms implemented in C++ using LEDA
The Current System

- Expert system written in Haskell
- Cluster code written in C++ using MPI
- Runs on the BCRI Compute Cluster
- Collection of approx. 200 theorems
- Approx. 180 algorithms implemented in C++ using LEDA
- Some debugging still needs to be done...
Possible Extensions

Screen for inconsistent sets of required properties
Possible Extensions

6. Screen for inconsistent sets of required properties

6. Allow for numeric comparisons:
   Looking for graph with $e \leq 4$
   Have theorem: $A \Rightarrow e = 2$
   System should infer property $A$ into wish list
Possible Extensions

1. Screen for inconsistent sets of required properties
2. Allow for numeric comparisons:
   Looking for graph with \( e \leq 4 \)
   Have theorem: \( A \Rightarrow e = 2 \)
   System should infer property \( A \) into wish list
3. Extend to deal with directed graphs
Screen for inconsistent sets of required properties

Allow for numeric comparisons:
Looking for graph with $e \leq 4$
Have theorem: $A \Rightarrow e = 2$
System should infer property $A$ into wish list

Extend to deal with directed graphs

Extend to other areas of discrete mathematics