What is a good diagram?
What is a good diagram? (reprise)

- A little history
- Some concepts
- Some observations, two of which are surprising
- A new metric
- A wild conjecture
- A nostalgic theorem

Peter Eades,
University of Sydney

With help from
- SeokHee Hong
- Karsten Klein
- Quan Nguyen
- Many others
Some concepts
What is a diagram?
The visualization researcher’s viewpoint:

- **Visualisation pipeline**

- A diagram is a graphical and geometric representation $V(d)$ of an element $d$ of a data set $D$.
- The diagram is perceived by a human.
- For this talk:
  - The data $d$ is a graph.
  - The visualization function $V$ is executed by a computer.
What is a *good* diagram?
What is a quality metric?
Ineffective
- Tangled, hard to read
- Does not communicate

Effective
- Gives insight
- Can communicate truth
This talk is about quality metrics for diagrams:

We want a quality metric $Q$:

$$Q: L \to R^k$$

where $L$ is the space of possible diagrams and $R^k$ is $k$-dimensional real space. (For simplicity we mostly assume $k = 1$.)

The metric $Q$ needs to have a number of mathematical properties, but the main requirement is something like this:

$v_1 \in L$ is a better diagram than $v_2 \in D$ if and only if

$$Q(v_1) > Q(v_2).$$
Why do we want to define quality metrics?
The constrained optimisation paradigm

- One of the earliest, biggest, and most pervasive success stories of Computer Science is the constrained optimisation strategy:

1. Find out what users want
2. Formalize what they want
   - Space $S$ of possible solutions
   - Space of $C$ constraints on $S$
   - Objective (quality) function $Q : S \rightarrow R$
3. Use an optimisation method to obtain what they want
   - Optimise $Q(s)$ over $s \in S$ subject to $C$

One of the most significant challenges in this strategy is defining the objective (quality) function $Q$.

- $Q$ must be formal and computable
- $Q$ must capture requirements as closely as possible
Why do we want to define quality metrics?

- To enable us to create visualization functions under the optimisation paradigm.
A little history
Quality metrics for graph visualization have been discussed since the 1970s.

- CCITT (1970s, 1980s)
- James Martin (1970s)
- Sugaya (1975)
- Sugiyama et al. (1978)
- Batini et al. (1985)
### 2.3.1 構造規則

<table>
<thead>
<tr>
<th>種類</th>
<th>図面表现</th>
<th>分類</th>
</tr>
</thead>
<tbody>
<tr>
<td>意</td>
<td>特定の頂点の系列を線路配置する</td>
<td>USLB</td>
</tr>
<tr>
<td>例</td>
<td>特定の頂点の系列を曲線配置する</td>
<td>USLB</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を区別配置する</td>
<td>UMLB</td>
</tr>
<tr>
<td>細</td>
<td>特定の線間の関係を線路の配置する</td>
<td>NTLB</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を曲線配置する</td>
<td>NTLB</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を区別配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を線路の配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を曲線配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を区別配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を線路の配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を曲線配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を区別配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を線路の配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を曲線配置する</td>
<td>NMLF</td>
</tr>
<tr>
<td>契</td>
<td>特定の線間の関係を区別配置する</td>
<td>NMLF</td>
</tr>
</tbody>
</table>

---

2.3.2 動画規則と分類

動画規則は意味的規則と構造的規則とに分けられる。

意味的規則は、頂点間の意味（頂点の重要度や頂点の関係の強さなど）から処理し、図面配置や配置規則は、構造情報のバイレベル情報による自動的抽出されたものである。重要な意味的規則を表2.3.1に示す。
These early metrics

- Based on *Intuition* and *Introspection* ($I^2$)
- Subjective
- Depend on the application domain
- Depend on many circumstances

- *Lacked scientific validation*

- *Lacked explicit formal definition*
Scientific validation
Purchase et al., 1997:

- **Edge crossings** negatively affect human readability:
  - More edge crossings means more human errors
  - More edge crossings means more human time spent

- Follow-up experiments (Ware et al. 2002, Huang et al. 2004) confirmed these results.

<table>
<thead>
<tr>
<th>Aesthetic Variation</th>
<th>few</th>
<th>some</th>
<th>many</th>
</tr>
</thead>
<tbody>
<tr>
<td>bends</td>
<td>6</td>
<td>18</td>
<td>30</td>
</tr>
<tr>
<td>crossings</td>
<td>6</td>
<td>24</td>
<td>42</td>
</tr>
<tr>
<td>symmetry</td>
<td>4.6</td>
<td>25.7</td>
<td>51</td>
</tr>
</tbody>
</table>

Fig. 3. Results for the dense graph
19 edge crossings

0 edge crossings
The overriding result of a number of human experiments:

- Edge crossings negatively affect readability of a graph visualization
- The most dominant geometric property of good graph visualization is the lack of edge crossings
Explicit formal definition
Formalise an edge crossing metric $Q_X$:

- We want to define a computable function $Q_X : L \rightarrow R$, such that higher value of $Q_X$ implies fewer crossings (and it’s nice if $0 \leq Q_X(D) \leq 1$).

We need a formal definition of quality to use as an optimisation goal to create visualizations.
Formalise an edge crossing metric $Q_X$:

- Suppose that $G$ is a graph with $m$ edges.
- Suppose that $D$ is a drawing of $G$.
- Suppose that $D$ has $X(D)$ edge crossings.

- Let $M = 0.5(m(m - 1) - \sum_u(d_u(d_u - 1)))$
  ($M$ is a naive upper bound on the possible number of edge crossings)

- Then formally define $Q_X(D)$:

\[
Q_X(D) = \frac{M - X(D)}{M}
\]

(that is, invert and normalize the number of crossings to $[0,1]$)
**Note:**
There are many ways to formalize the edge crossing metric.
- The maximum number of crossings on an edge, over all edges,
- The maximum/average number of mutually crossing edges
- Depends on whether edges are straight lines
- Depends on whether edges are allowed to cross themselves

See
Optimization methods are used to construct visualizations of graphs with few edge crossings

**Minimum Crossing Problem**

Find a topological embedding of an input graph $G$ with as few edge crossings as you can.

- NP-hard optimization problem
- Naïve methods give poor results
Minimum Crossing Problem
Find a topological embedding of an input graph $G$ with as few edge crossings as you can.

Optimization method (1980s – 2000s):

1. Find a maximum planar subgraph $G' = (V, F)$, with $F \subseteq E$; find a topological embedding of $G'$

2. Route the edges of $E - F$ using a shortest path through the faces, replacing crossing points with dummy nodes, making a planar graph $G''$

3. Find a topological embedding of $G''$
Use integer linear programming

Variables
\[ x_e \forall \text{edge } e \]

Objective
\[ \text{maximize } \sum x_e \]

Constraints
\[ 0 \leq x_e \leq 1 \quad \forall \text{edge } e \in E \]
\[ \sum_{e \in K} x_e \leq |K| - 1 \quad \forall \text{Kuratowski subgraph } K \text{ in } G \]
\[ \sum_{e \in E} x_e \leq 3|V| - 6 \]

\[ x_e \in \{0, 1\} \]
An observation
An observation:

The minimum crossing approach works well, and is beloved by graph drawing researchers.
Examples
Examples
Another observation
Another observation:

Data sets are now very large.
How many edge crossings can you see?
A surprising observation
A surprising observation:

No commercial visualization software uses crossing minimization\(^1\).

\(^1\) Note: This statement is actually false, but it is almost true.
Some more concepts
Readability and Faithfulness
Data space

Data

Data space

Picture

$V$ is a **visualization** function $V: D \rightarrow L$ that assigns a picture $V(d) \in L$ for each $d \in D$.

Geometric space

Visualization function

$L$

Human

$P$ is a **perception** function $P: L \rightarrow K$ that assigns knowledge $P(l) \in K$ for each $l \in L$.

Knowledge space

Perception function

$K$

Task

$T_K$ is a **task** function $T_K: K \rightarrow R$ that assigns a result $T_K(k) \in R$ for each $k \in K$.

Result space

$R$
\( T_D \) is a **data task** function \( T_D : K \rightarrow R \) that assigns a result \( T_D(d) \in R \) for each \( d \in D \).

\( T_L \) is a **picture task** function \( T_L : L \rightarrow R \) that assigns a result \( T_L(l) \in R \) for each \( l \in D \).
**Task**

- **Data**
  - Data space
  - $D$

- **Picture**
  - Geometric space
  - $V$
  - Visualization function
  - $L$

- **Human**
  - Knowledge space
  - $P$
  - Perception function
  - $K$

**Task functions**

- $T_D$ is an **oracle**:
  - given the data, $T_D$ always returns the correct result

- $T_L$ is an **oracle**:
  - given the picture, $T_L$ always returns the correct result

- $T_K$ is a Task function

**Result space**

- $R$
Faithfulness

a) Information faithfulness
b) Task faithfulness
c) Change faithfulness
Information faithfulness

The visualization function $V$ is *information faithful* if and only if:

- $V$ is one-to-one; that is,
- If $V(d_1) = V(d_2)$ for $d_1, d_2 \in D$, then $d_1 = d_2$; that is,
- $V$ has an inverse function $V^{-1}$; that is,
- (informally) the picture $V(d)$ of a data item $d$ has the exactly same information as the data item $d$.

“…a **faithful representation** $\rho$ of a group $G$ on a vector space $V$ is a linear representation in which different elements $g$ of $G$ are represented by distinct linear mappings $\rho(g)$ … “

(Wikipedia)
Task faithfulness

Suppose that \( T = (T_D, T_L, T_K) \) is a task function.

The visualization function \( V \) is *task faithful for* \( T \) if:

1. \( T_L(V(d)) = T_D(d) \) for all \( d \in D \); that is,

2. (informally) The picture task oracle \( T_L \) on the picture \( V(d) \) of a data item \( d \) returns the same result as the data task oracle \( T_D \) on the data item \( d \).
Change faithfulness

We use distance functions $\Delta_D$ and $\Delta_L$ on the data space $D$ and the geometric space $L$ respectively.

The visualization function $V$ is *change faithful* if

- $\Delta_D(d_1, d_2) = \Delta_L(V(d_1), V(d_2))$ for all $d_1, d_2 \in D$; that is,
- the visualization function $V$ is an isometric transformation; that is,
- (informally) The change in the picture is proportional to the change in the data.
Faithfulness concepts: summary

a) Information faithfulness: $V$ is one-to-one.
b) Task faithfulness: The picture task oracle on $V(d)$ gives the same result as the data task oracle on $d$.
c) Change faithfulness: $V$ is an isometric transformation.

**Informal conjecture:**
A graph visualization function is good if and only if it is

- Information faithful,
- Task faithful for the tasks at hand, and
- Change faithful,

and

- Information readable,
- Task readable,
- Change readable.
Some examples to illustrate faithfulness
Many methods are information faithful on small graphs.

Information faithfulness implies task faithfulness for all tasks.
Edge bundling methods are not information faithful, because many graphs map to the same picture.

They sacrifice faithfulness and gain readability

They are task faithful for some tasks, eg tasks about identifying major flows.
- Stress / force – based methods are not information faithful, because nodes can overlap.
- However, they seem to be task faithful for tasks that based on graph-theoretic distances
  - eg, identifying clusters
<table>
<thead>
<tr>
<th><strong>Faithfulness requirements</strong></th>
<th><strong>Readability requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information</strong></td>
<td></td>
</tr>
<tr>
<td>All the information in the data is in the picture</td>
<td>All the information in the picture is in the user’s knowledge</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td></td>
</tr>
<tr>
<td>There is enough information in the picture to perform the task correctly</td>
<td>The user’s task performance is consistent with the information in the picture.</td>
</tr>
<tr>
<td><strong>Change</strong></td>
<td></td>
</tr>
<tr>
<td>The amount of change in the picture is proportional to the amount of change in the data</td>
<td>The mental map is preserved.</td>
</tr>
</tbody>
</table>
**Intuition**

For large diagrams, faithfulness is commercially significant.
Life is not Boolean …..

We want to make statements such as:

- “This visualization function is *fairly faithful*.”
- “*This* visualization function is *more faithful* than *that* visualization function.”

Big data requires
- Clustering of nodes
- Clustering of edges

Big data makes 100% faithfulness impossible

- *We need faithfulness metrics*
### Faithfulness
- Metrics not well developed

### Readability
- Metrics well developed

---

We hope to develop visualization methods as algorithms that optimise faithfulness and readability.
Faithfulness metrics
**Stress**

The most obvious candidate for a faithfulness metric is **stress**:  
- Define a stress value for each pair \(u, v\) of nodes. 
- Find a layout that minimizes stress over the whole drawing.

For example
- For a pair \((u, v)\), stress \(\sigma(u, v) = w_{uv} | \Delta_D(p_u, p_v) - \Delta_G(u, v)|^2\)  
  (i.e., error from ideal distance) 
- For the whole graph, stress = \(\sum_{u,v} \sigma(u, v)\)

where
- \(\Delta_D(p_u, p_v)\) is the Euclidean distance in the drawing \(D\) between the location \(p_u\) of \(u\) and the location \(p_v\) of \(v\), 
- \(\Delta_G(u, v)\) is the ideal (or graph theoretic) distance in between \(u\) and \(v\), 
- \(w_{uv}\) is a weight factor.
Stress-based visualization methods have a long history

- Tutte’s theorem (1960)
- Multidimensional scaling methods (from 1960s)

Many stress “minimization” methods are available:

- Some are naive
- Some are sophisticated
- Some are effective
- Some are useless
- Some are quick
- Some are slow
Intuition:

- Crossings measures readability
- Stress measures faithfulness

Perhaps:

- The lack of commercial interest in edge crossings is because such methods are not task faithful
- The high commercial interest in stress-based methods is because such methods are task faithful for a wide variety of tasks.
Another surprising observation
The GION experiment
The GION experiment*

- GION is a specific interaction technique for graphs on large (wall-size) displays.
- We ran HCI-style experiments to test GION
- Subjects “untangled” large graphs using two different interaction techniques.
- Result: GION is faster than the standard technique.

The GION experiment data set: RNA Sequence Networks

Read alignment overlap graphs:
- Large
- locally dense
- Globally sparse
- Well structured
- Thousands of nodes
\( t = 0 \)
(Fruchterman-Reingold layout)

\( t = 6 \)
(after 6 seconds of user interaction)

\( t = 12 \)
(after 12 seconds of user interaction)
Surprising observation:

- On average, users increased both crossings and stress in untangling the graphs.
Perhaps:

- Stress is not a good metric for faithfulness.
- Crossings are not a good metric for readability (for large graphs).
A new metric
A new faithfulness metric

Intuition:
- In a good layout of a graph \( G \), the shape of the set of node locations is very similar to \( G \).
- In a bad layout of \( G \), the shape of the set of node locations is very different from \( G \).

1. Actually, it is not 100% new … something similar, called “precision of preservation of neighbours” is defined in the papers by Gansner, Hu amd Krishnan on the COAST method.
Background: the shape of a set of points in 2D as a graph
Alpha-shape (Edelsbrunner, Kirkpatrick & Seidel, 1983)

- Given a point set $S$, a point $p \in S$ is $\alpha$-extreme if there exists an empty open disk of radius $\alpha$ with $p$ on its boundary.
- Two points $p, q \in S$ are $\alpha$-neighbors if they share such an empty disk.
- The $\alpha$-shape of $S$ is the geometric graph whose vertices are the $\alpha$-extreme points and whose edges connect the respective $\alpha$-neighbors.

Alpha-shapes give graphs that capture the structure of the boundary of a set of points, but do not model the internal structure of the set of points.

So we consider proximity graphs, or beta-shapes →
Proximity graphs as shapes (Toussaint, 1988)

- “Join the dots” graph on a set of points in 2D
- Given a point set $S$, join two points $p, q \in S$ if $p$ and $q$ are “close” to each other.
- Capture the shape of a set of points as a graph.

Examples

- Euclidean minimum spanning tree
- Nearest neighbour graph: join $p, q \in S$ if $d(p, q) \leq d(p, q')$ for all $q' \in S$.
- $k$-nearest neighbour graph: join $p, q \in S$ if there are at most $k$ points $q' \in S$ for which $d(p, q) \geq d(p, q')$.
- Relative neighbourhood graph
- Gabriel graph
- $\beta$-shape ($\beta$-skeleton)
Point set → MST → N3NG
Proposed faithfulness quality metric $Q_P$:
- The similarity between the graph and a proximity graph on the shape of the set of node locations

$$Q_P(V(G)) = \text{similarity between } G \text{ and } N(F(V(G)))$$
A bad layout $D_0$ of a graph $G$

The Euclidean Minimum spanning tree of the node locations in $D_0$
A slightly better layout $D_1$ of the same graph $G$.

The Euclidean Minimum spanning tree of the node locations in $D_1$. 
An even better layout $D_2$ of the same graph $G$

The Euclidean Minimum spanning tree of the node locations in $D_2$
An really good layout \( D_3 \) of the same graph \( G \)

The Euclidean Minimum spanning tree of the node locations in \( D_3 \)
How can you measure the similarity of two graphs?
Jaccard similarity measure between two nodes

If \( u \) is a node in \( G \) and \( u' \) is the corresponding node in \( G' \), then

\[
J(u, u') = \frac{|N(u) \cap N(u')|}{|N(u) \cup N(u')|}
\]

where \( N(u) \) is the set of neighbours of \( u \) in \( G \) and \( N(u') \) is the set of neighbours of \( u' \) in \( G' \).

Note:

- \( 0 \leq J(u, u') \leq 1 \)
- If \( u \) and \( u' \) have very similar sets of neighbours, then \( J(u, u') \) is large
- If \( u \) and \( u' \) have very different sets of neighbours, then \( J(u, u') \) is small
If \( u \) is a node in \( G \) and \( u' \) is the corresponding node in \( G' \), then

\[
J(u, u') = \frac{|N(u) \cap N(u')|}{|N(u) \cup N(u')|}
\]

where \( N(u) \) is the set of neighbours of \( u \) in \( G \) and \( N(u') \) is the set of neighbours of \( u' \) in \( G' \).

**Jaccard sum similarity measure** \( JS(G, G') \) of two graphs:

\[
JS(G, G') = \sum_u J(u, u')
\]

where the sum is over all vertices \( u \) in \( G \) with corresponding node \( u' \) in \( G' \).
$JS(G, G') = 0.1505$

$JS(G, G') = 0.2781$

$JS(G, G') = 0.4436$

$JS(G, G') = 0.5041$
The GION experiment and the Jaccard sum similarity metric
Initial drawing $D_0$

Minimum spanning tree $MST_0$ of initial drawing

$t = 0$

$JS = 0.003099$
Drawing $D_3$

Minimum spanning tree $MST_3$ of $D_3$

$JS = 0.003511$
\( t = 6 \)

Drawing \( D_6 \)

Minimum spanning tree \( MST_6 \) of \( D_6 \)

\[ JS = 0.004078 \]
$t = 9$

Drawing $D_9$

Minimum spanning tree $MST_9$ of $D_9$

$JS = 0.004369$
\( t = 12 \)

Drawing \( D_{12} \)

Minimum spanning tree \( MST_{12} \) of \( D_{12} \)

\[ JS = 0.004435 \]
Untangling increases Jaccard sum similarity
A wild conjecture
Conjecture
The Jaccard sum similarity metric is a good quality measure for large graph drawings.

Note: there are many possible variations
- Use \textit{Relative Neighbourhood Graph} or \textit{Gabriel Graph} as the proximity graph.
- Use a different measure of the similarity between two graphs.
A nostalgic theorem
Optimization Problem
- Suppose that the shape function is minimum spanning tree
- Suppose that the quality function is the Jaccard sum distance $JS$

**High-Faithfulness Drawing Problem**

Input: Graph $G$

Output: A location $p(v)$ for each node $v$ of $G$ such that $JS(G, MST(P))$ is maximal, where $P = \{p(v) : v \text{ in } G\}$.

**Theorem**
The High-Faithfulness Drawing Problem is NP-hard.
In fact, the High-Faithfulness-Drawing problem has the following sub-problem:

**MST Realisation Problem**
Input: Graph $G$
Output: A location $p(v)$ for each $v$ in $G$ such that $G = \text{MST}(P)$, where $P = \{p(v) : v \in G\}$.

**Nostalgic Theorem**
The MST Realisation Problem is NP-hard.

The logic engine

- A mechanical device that simulates in instance of NAE3Sat
- Can be made *flat* if and only if the NAE3Sat instance has a “yes” solution.

Not-All-Equal-3-Sat (NAE3Sat)

**Instance**: A set $C$ of clauses over a set $V$ of variables

**Question**: Is there a truth assignment $t: V \to \{True, False\}$ such that every clause $c \in C$ has at least one *True* and at least one *False* literal?

A well-known NP-complete problem.
The logic engine

- A mechanical device that simulates in instance of NAE3Sat
- Can be made flat if and only if the NAE3Sat instance has a “yes” solution.

The logic engine consists of

- A frame
- An axle
- A rod for each variable; the rod is attached to the axle
- A lot of flags attached to the rods

Parts can move

- The rods swing around the axle
- The flags swing around the rods
The frame
The frame
Axle

A rod

The frame

Axle
Each rod can turn about the axle

The frame
There are many flags
Each flag can turn freely about its rod
A logic engine has many configurations, depending on

- How each flag is turned around its rod
- How each rod is turned around the axle.

A configuration is **flat** if

- No flag hits another flag, and
- No flag hits the frame.
Flat configuration
Not-flat configuration
Note: If there is a row that is full of flags, then a flat configuration is impossible.
Note: If there is a row that is full of flags, then a flat configuration is impossible.

In fact, there is a flat configuration if and only if the rods can be turned so that there is a vacant position in each row of flags.
Mapping an instance $c, V$ of NAE3Sat to the logic engine

- Each variable $x_i \in V$ corresponds to the $i$th rod of the logic engine
  - Turning the $i$th rod about the axle swaps between $true$ and $false$ values of $x_i$. 
Rods represent variables

$x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$
Mapping an instance $c, V$ of NAE3Sat to the logic engine

- Each variable $x_i \in V$ corresponds to the $i$th rod of the logic engine
  - Turning the $i$th rod about the axle swaps between $true$ and $false$ values of $x_i$.

- For each clause $c \in C$, we have
  - A row of flags above the axle, and
  - A row of flags below the axle.
Rods represent variables, rows of flags represent clauses
Mapping an instance $c, V$ of NAE3Sat to the logic engine

- Each variable $x_i \in V$ corresponds to the $i$th rod of the logic engine
  - Turning the $i$th rod about the axle swaps between $true$ and $false$ values of $x_i$.

- For each clause $c \in C$, we have
  - A row of flags above the axle, and
  - A row of flags below the axle.

  - Above the axle
    - we place a flag on rod $i$ in row $j$ if $x_i$ does not occur in $c_j$.

  - Below the axle
    - we place a flag on rod $i$ in row $j$ if $\overline{x_i}$ does not occur in $c_j$. 
Above the axle: we place a flag on rod $i$ in row $j$ if $x_i$ does not occur in $c_j$.

Below the axle: we place a flag on rod $i$ in row $j$ if $\bar{x}_i$ does not occur in $c_j$. 
\[ c_1 = (x_4 \lor \overline{x_2} \lor \overline{x_3}), \quad c_2 = (x_2 \lor x_4 \lor \overline{x_2}), \quad c_3 = (x_4 \lor \overline{x_1} \lor \overline{x_3}) \]
Mapping an instance $c, v$ of NAE3Sat to the logic engine

- Turning the $i$th rod about the axle swaps between $true$ and $false$ values of $x_i$.
- Turning the $i$th rod about the axle chooses a truth assignment for each variable.

Thus

- The logic engine has a flat configuration
  
  *if and only if*

- We can turn the rods so that each row of flags has a vacant position
  
  *if and only if*

- The NAE3Sat instance has a truth assignment with at least one true and at least one false literal in each clause.

Thus

- Finding a flat configuration of a logic engine is NP-hard
**Nostalgic Theorem**
The MST Realisation Problem is NP-hard.

Proof

- Define a tree $T$ such that every MST layout of $T$ looks like a logic engine
High-Faithfulness Drawing Problem

Input: Graph $G$

Output: A location $p(v)$ for each node $v$ of $G$ such that $JS(G, MST(P))$ is maximal, where $P = \{p(v): v \in G\}$.

Thus the High-Faithfulness Drawing Problem is NP-hard.

Open problem: how can we efficiently approximate solutions to the High-Faithfulness Drawing Problem?
Preference experiment

- A recent in-progress experiment
- Aim: to determine geometric properties of graph visualizations that people like.

Preliminary result:
- There seems to be a slight preference for diagrams with lower stress and fewer crossings
Final remarks
What this talk was about:

1. I think we need to measure the faithfulness of large graph visualization.

2. I suggest that the Jaccard sum similarity is a good candidate for such a metric.

3. I don’t know how to optimise this metric.