# Drawing graphs with few lines

Theo Doukas

02.06.2016

# Graphs and Drawings

## Graph basics

- $G = (V, E)$ (*Vertices* and *Edges*)
- *Simple* graph: At most one edge between two vertices; edges bear no direction, i.e.

$$E \subseteq \{\{u, v\} \mid u, v \in V\}.$$

We will only look at simple graphs.

# Graphs and Drawings

## Drawing (also, *"Embedding"*)

- Mapping $\varphi$ into a drawing area $S \subseteq \mathbb{R}^2$ (usually, a rectangle)

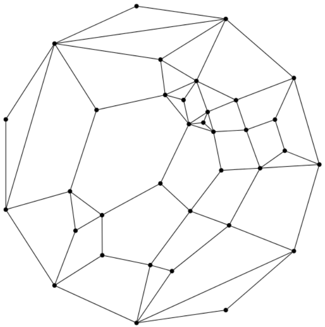
$$\begin{array}{rcll} v \in V & \mapsto & \varphi(v) \in S & \text{(injective)} \\ e = \{u, v\} \in E & \mapsto & \varphi(e) = \gamma_e : [0,1] \to S & \text{(continuous)} \\ & \text{where} & \gamma_e(0) = u,\ \gamma_e(1) = v & \end{array}$$

We will draw edges as straight lines.

# How to draw a graph?

## Analytic solution: Tutte, 1963

- Select (at least) three vertices and fix their positions
- Map the remaining vertices' positions to the barycentre of their respective neighbours
- Yields a linear equation system
- Works quite well, but has issues ($\rightarrow$ poor vertex resolution).

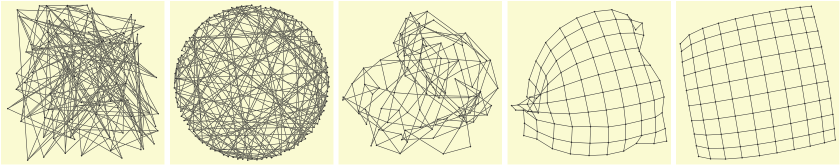


# How to draw a graph?

## Numeric, iterative solution: Eades, 1984; Fruchterman & Reingold, 1991 (and many others)

- Drawing: result of a simulation of a physical system
- Attractive forces along edges, repulsive forces between vertices $\rightarrow$ *Force directed* algorithms
- "Temperature" $=$ Simulation step width, slowly decaying
- Meets aesthetical criteria.

### Example



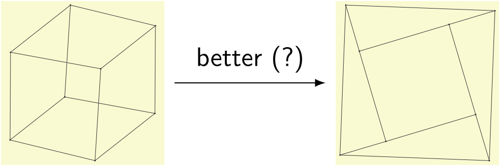
# Good drawings are aesthetically appealing

## Established criteria ([Eade84], [FrRe91], . . . )

- Visualize symmetries
- Avoid crossing edges
- Evenly distribute vertices

## Schulz, 2015: Minimal visual complexity

- Drawings that consist of fewer geometric primitives should be *easier to perceive*:




$\boxed{\rightarrow \text{straight paths}}$

# Introducing rigid / straight paths

## Problem

- How to create drawings that feature minimal visual complexity

## Idea

- Modify a force-directed algorithm to produce straight paths
- Do this by defining additional forces / constraints along paths

# How force-directed algorithms work

## Method by Fruchterman & Reingold (1991)

- Forces: $f_a(d) = \frac{d^2}{k}$, $f_r(d) = \frac{k^2}{d}$.
- Total force on vertex v:

$$F_v = \sum_{w \neq v} f_a(d_{vw}) \cdot e_{wv} + \sum_{\{v,w\} \in E} f_r(d_{vw}) \cdot e_{vw}$$

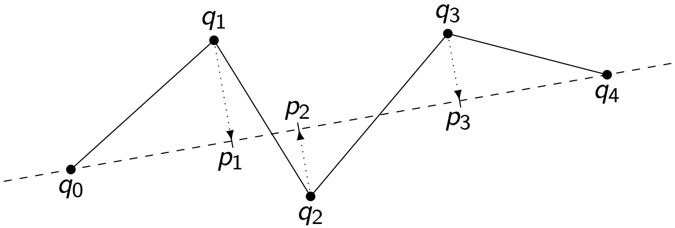
  where $d_{vw} := \|\varphi(v) - \varphi(w)\|$, $e_{vw} := (\varphi(v) - \varphi(w))/d_{vw}$.
- Displacement:

$$\Delta(v) = \min\{\|F_v\|, T\} \cdot \frac{F_v}{\|F_v\|}$$

- $n$th Iteration: $\varphi_{n+1}(v) := \varphi_n(v) + \Delta_n(v)$.

# Modifications

## The *straight* model



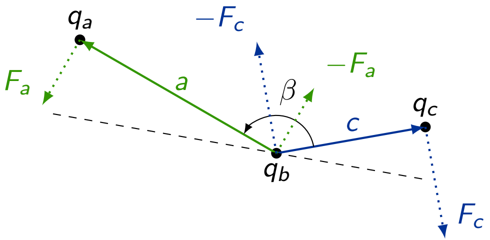

- Positions (projects) vertices along a straight line:

$$p_i = q_0 + \lambda_i (q_n - q_0) \quad \text{where} \quad \lambda_i := \frac{\langle q_i - q_0,\; q_n - q_0 \rangle}{\|q_n - q_0\|^2}$$

- Introduces *constraints*

# Modifications for straight paths

## The *convex* model




- Attempts to align adjacent edges:

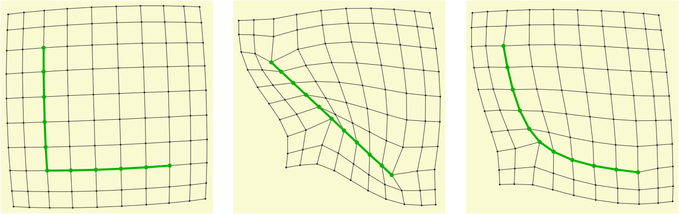
$$\beta \to \pi, \quad \|F_{a,c}\| = \text{const} \cdot |\beta - \pi|$$

- Introduces *forces*

# Applying the models

## Example

- Visualization: Applied on L-shaped path inside a grid

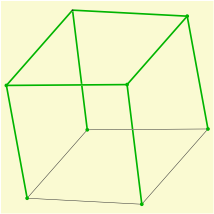


(no forces) straight model convex model

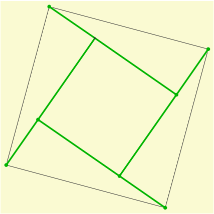
# Applying the models

## Example

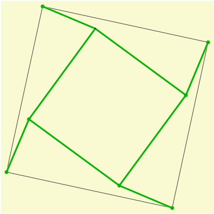
- Applied on the cube graph



(no forces)



straight model



convex model

## That's fine, but ...

- How do we choose the paths?

# Choosing paths

## Choose randomly

- Does in fact reduce visual complexity ...
- ... but creates overlapping edges.

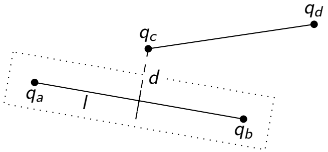
## Avoiding overlapping edges

- Iterative approach
- Try paths, then judge quality of drawing
- Be able to undo steps ($\rightarrow$ backtracking)

# Judging the quality

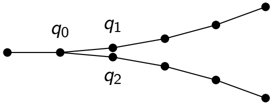
## Detect overlapping edges

- Small angle between edges
- $l, d$ within given intervals




## Identify edges which belong together

- Small angle between adjacent edges
- No unique partitioning




## Measure of quality

- $Q = -C - \text{const} \cdot v$
  ($C$ = num. geometrical elements, $v$ = num. overlaps)

# The algorithm

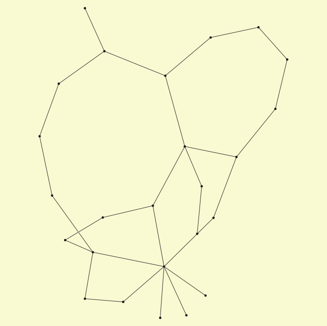
## Elementary operations

1. Create a path of minimal length $\ell = 2$
2. Extend a path by an adjacent edge
3. Join two paths (on a common boundary vertex)

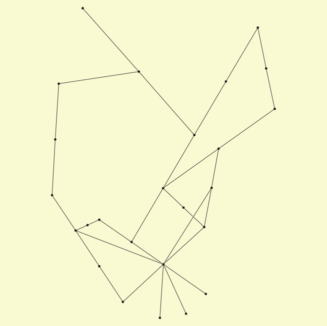
## Procedure

1. Try operation 1, judge quality; worse $\rightarrow$ undo. Three consecutive, unsuccessful attempts $\rightarrow$ step 2.
2. Same with operation 2. If successful: go back to step 1. Three unsuccessful attempts $\rightarrow$ step 3.
3. Same with operation 3. Success $\rightarrow$ step 1. Three times no success $\rightarrow$ End.

# Results

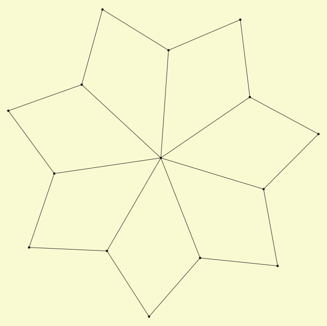


(a) $C = 29, \ v = 2$

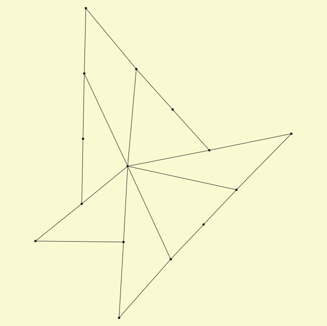


(b) $C = 16, \ v = 0$

# Results

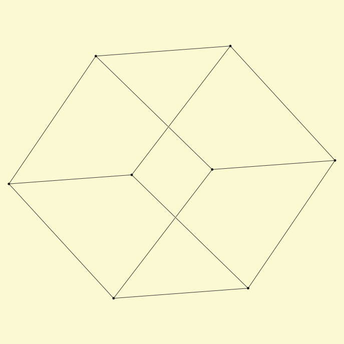


(c) $C = 21, \ v = 0$

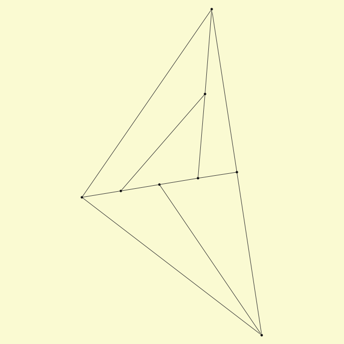


(d) $C = 9, \ v = 0$

# Results



(e) $C = 12,\ v = 0$



(f) $C = 7,\ v = 0$

# Summary

## Force-directed algorithms

- Standard approach for a wide range of applications
- Meet aesthetical criteria

## Novel idea: reduce visual complexity [Schu15]

- Modify a force-directed algorithm by introducing constraints / forces that create straight, rigid paths

## Choose rigid paths

- Iterative, trial and error approach

# Conclusion

## Results

- Examples: visual complexity was reduced to $\sim 50\%$.
- The algorithm places an outer loop around the drawing algorithm, increasing computation time by a factor $\sim 100$.
- Apparent tradeoff: Symmetry vs. complexity

## Future work

- Are the drawings produced indeed easier to perceive? This has yet to be investigated.

# References


Eades, Peter: A heuristics for graph drawing. *Congressus numerantium*, 1984, 42. Jg., S. 146–160.

Fruchterman, Thomas M.J.; Reingold, Edward M.: Graph drawing by force-directed placement. *Software: Practice and Experience*, 1991, 21. Jg., Nr. 11, S. 1129–1164.

Schulz, André: Drawing Graphs with few arcs. *Journal of Graph Algorithms and Applications*, 2015, 19. Jg., Nr. 1, S. 393–412.

Tutte, William T.: How to draw a graph. *Proceedings of the London Mathematical Society*, 1963, 13. Jg., Nr. 3, S. 743–768.