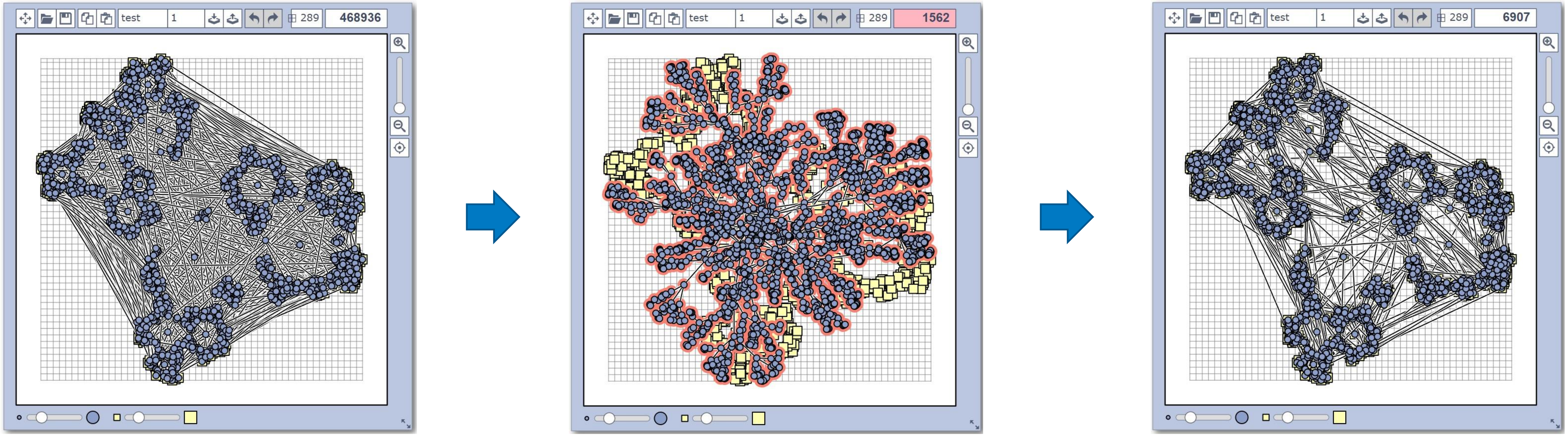


Minimizing crossings in point-set embeddings of graphs

Colloquium

Table of contents

Introduction

- **Target:** Participation in the Graph Drawing Contest (GDC) 2024
  - Symposium on Graph Drawing and Network Visualization

- **Challenge of 2023 & 2024:** Minimizing crossings in point-set embeddings of graphs
  - Larger (referred to as Automatics) & smaller (referred to as Manuals) instances
  - Provided time limit of 60 minutes

- Build on a previous Bachelor thesis by Alexander Kutscheid
  - Combined approach of…
    1. Force-directed algorithms (FDA)
    2. Greedy assignments
    3. Simulated Annealing (SA)

Definitions

Definition 1: A <u>curve</u> is a multi-subset of $\mathbb{R}^2$ of the form $\alpha = \{\gamma(x) : x \in [0,1]\}$, where $\gamma: [0,1] \to \mathbb{R}^2$ is a continuous mapping from the closed interval $[0,1]$ to the plane. $\gamma(0)$ and $\gamma(1)$ are called endpoints.

Definition 2: A curve is a <u>straight line segment</u>, if it can be defined as $\gamma(x) = (1-x) \cdot \gamma(0) + x \cdot \gamma(1)$. Thus, the function $\gamma(x)$ is a “linear interpolation” between the endpoints.

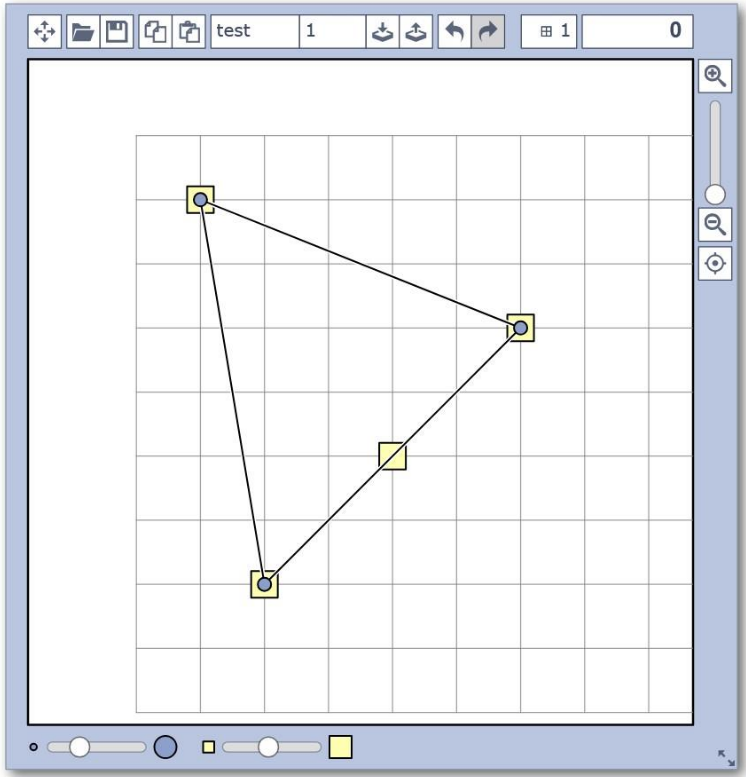
- In accordance with this definition, straight lines can be defined with vectors, whereby $x \in [0,1]$:
  $\gamma(x) = (1-x) \cdot \gamma(0) + x \cdot \gamma(1) = \gamma(0) + x \cdot \big(\gamma(1) - \gamma(0)\big) = \gamma(0) + x \cdot \overrightarrow{\gamma(0)\gamma(1)}$

Definitions

Definition 3: A <u>drawing</u> Γ of a graph $G(V, E)$ maps each vertex $v \in V$ to a distinct point $\Gamma(v)$ of a plane in $\mathbb{R}^2$ and each edge (u, v) to a curve $\Gamma(u, v)$ with the endpoints $\Gamma(u)$ and $\Gamma(v)$.

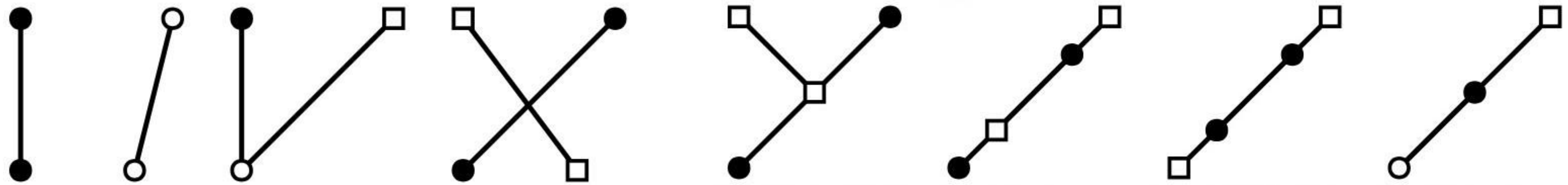
- It is assumed, that $\Gamma: V \rightarrow \mathbb{R}^2$ is always injective.

Definition 4: A drawing is a <u>point-set embedding</u> (PSE) of $G(V, E)$ on a finite point-set $P \subseteq \mathbb{R}^2$, such that the vertices' mapping can be restricted to $\Gamma : V \rightarrow P$ and all curves are drawn as straight lines.



Definitions

$$Score(\Gamma) = \sum_{i=1}^{|E|} \sum_{j=i+1}^{|E|} Cross(e_i, e_j)$$



Case 1a: no common point, so these two edges have a **cross** value of **0**.

Case 1b: the only common point is the shared endpoint of both edges (white circle), so these two edges have a **cross** value of **0**.

Case 2: the only common point is interior to both edges, so these two edges have a **cross** value of **1**.

Case 3a: the lower white square lies on top of the edge connecting the black dots, so these two edges have a **cross** value of $\boldsymbol{n}$.

Case 3b: the lower white square lies on top of the edge connecting the black dots (and the upper black dot lies on top of the edge connecting the white squares), so these two edges have a **cross** value of $\boldsymbol{n}$.

Case 3c: both black dots lie on top of the edge connecting the white squares, so these two edges have a **cross** value of $\boldsymbol{n}$.

Case 3d: though the two edges share a vertex (white circle) the black dot lies on top of the edge connecting the shared vertex to the white square, so these two edges have a **cross** value of $\boldsymbol{n}$.

Approaches: Score calculation – Excessive counting

Input: Drawing Γ of a graph $G(V, E)$
Output: Score of Γ

$score \leftarrow 0$
foreach $\{e_1, e_2\}$ **in** $\binom{\Gamma(E)}{2}$ **do**
  if e_1 and e_2 have no endpoint in common **then**
    if an endpoint of e_1 lies on e_2 **or** vice versa **then**
      $score \leftarrow score + |V|$ ⬅ Fall 3a, 3b, 3c
    else if e_1 and e_2 cross **then**
      $score \leftarrow score + 1$ ⬅ Fall 2
  else
    if the unshared endpoint of e_1 lies on e_2 **or** vice versa **then**
      $score \leftarrow score + |V|$ ⬅ Fall 3d
return $score$

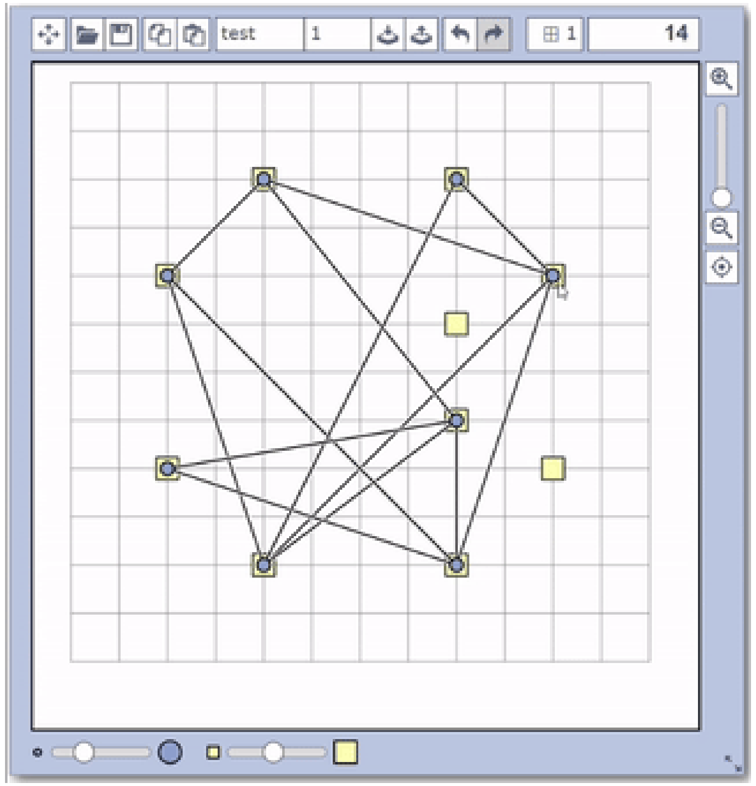
- Calculations require at least $\binom{|E|}{2} = \frac{|E| \cdot (|E|-1)}{2}$ comparisons.

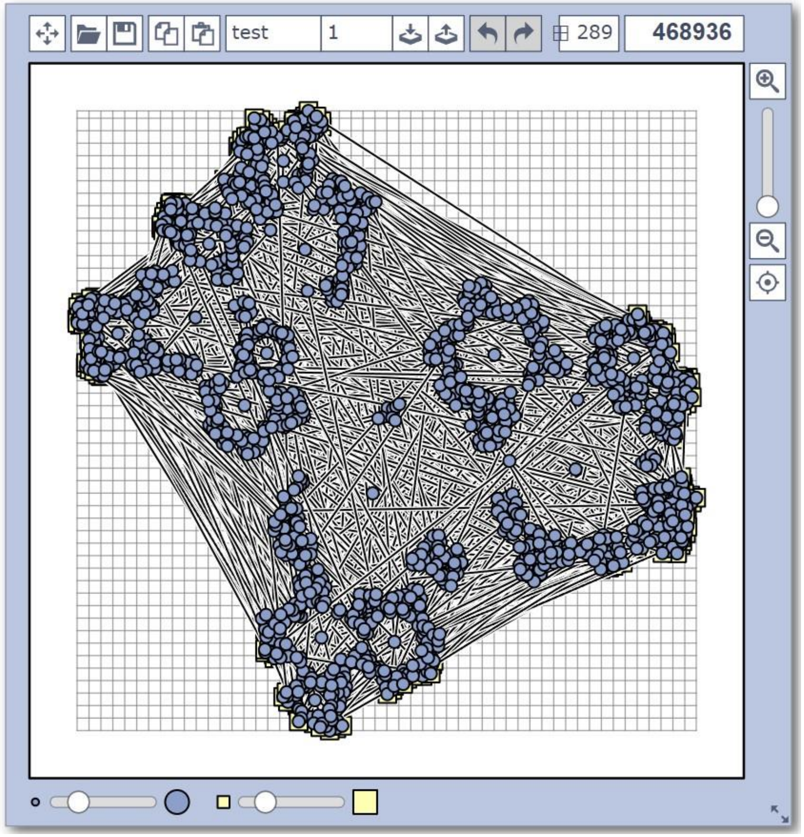
Approaches: Score calculation – Lazy counting

- Tracked score, which is updated on each modification.
- Only the impact of adjacent edges (before and after) need to be recalculated, if a vertex is to be repositioned.
- Introduction of a penalty operator:

$$pen(v) = \sum_{(v,u)\in E} \sum_{e\in E\setminus(v,u)} Cross\big((v,u), e\big)$$

- An update requires only $2 \cdot (deg(v) \cdot |E|)$ comparisons.



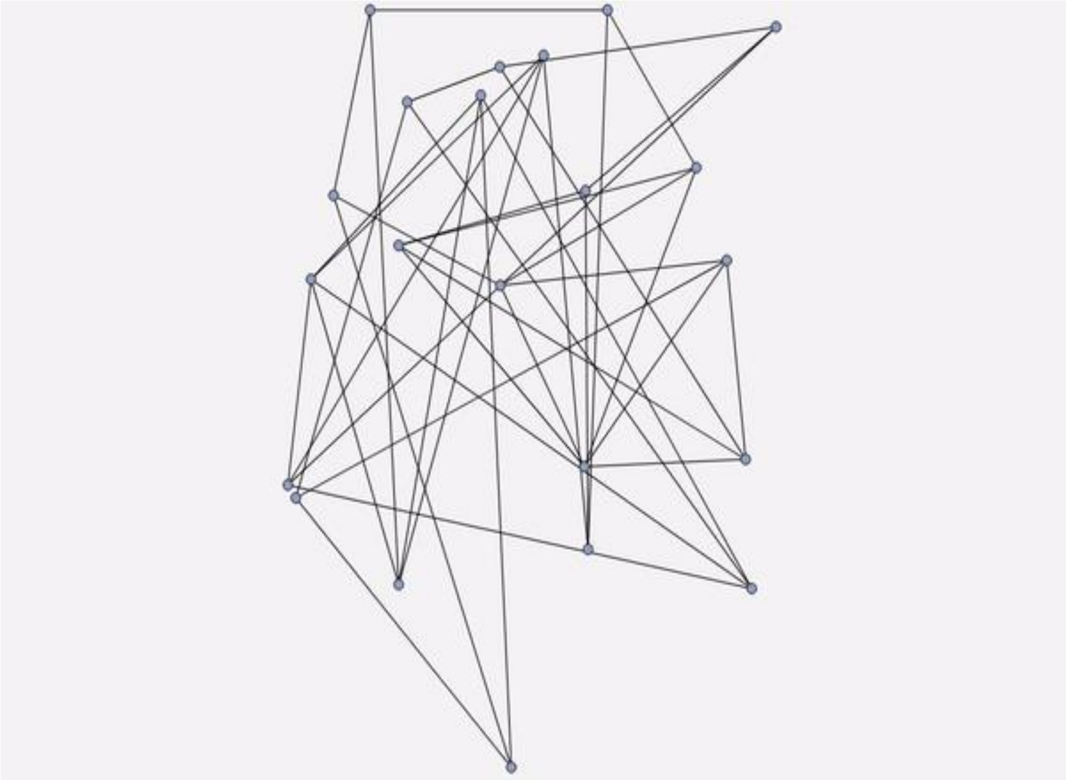


Input

Approaches: FDAs

- <u>Force-directed algorithms</u> aka spring embedders
- “*To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system [...]. The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state*”[1] – Peter Eades

- Application of repelling (f_{repl}: $V \times V \rightarrow \mathbb{R}^2$) and attracting ($f_{attr}$: $V \times V \rightarrow \mathbb{R}^2$) forces.
- Cautious Cooling ($\delta_t : \mathbb{R}^2 \rightarrow \mathbb{R}^2$) further ensures stabilization in late iterations



Approaches: FDAs

Input: Drawing Γ of a graph $G(V, E)$
Output: Beautified variant of Γ

$t \leftarrow 1$
while $t < T$ **and** $\max_{v \in V} ||\delta_t(F(v))|| > \varepsilon$ **do**
  foreach v **in** V **do**
    $F(v) \leftarrow \sum_{u \in V} f_{rep}(v, u) + \sum_{(v,u) \in E} f_{attr}(v, u)$
  foreach v **in** V **do**
    $\Gamma(v) \leftarrow \Gamma(v) + \delta_t(F(v))$
  $t \leftarrow t + 1$
return Normalize(Γ)

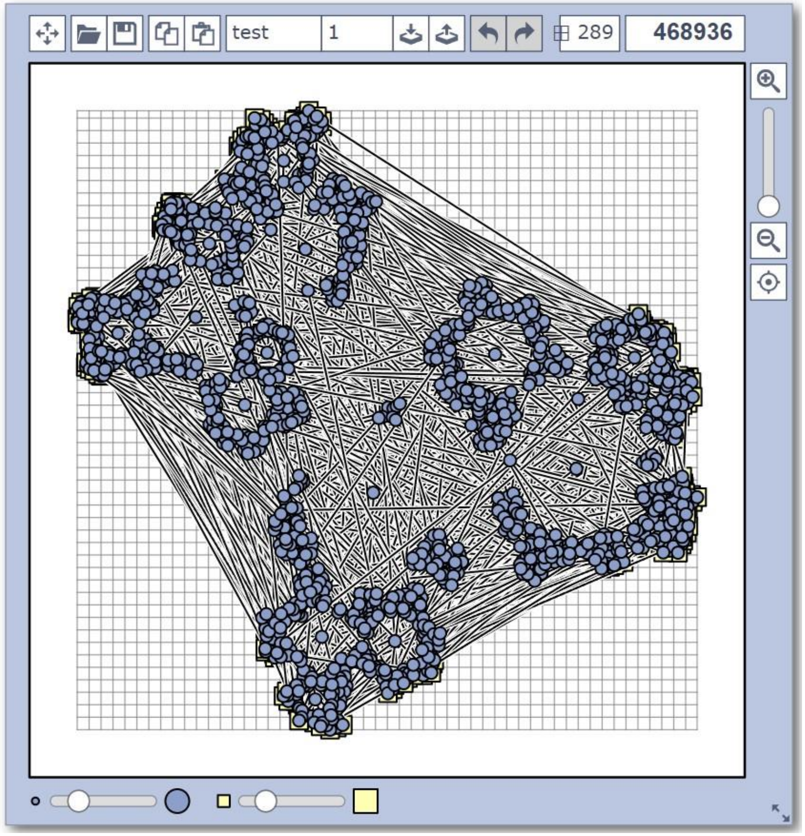
- Fixed value: $\varepsilon = 0.0001$
- Subsequent min-max normalization rescales the drawing to the borders.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

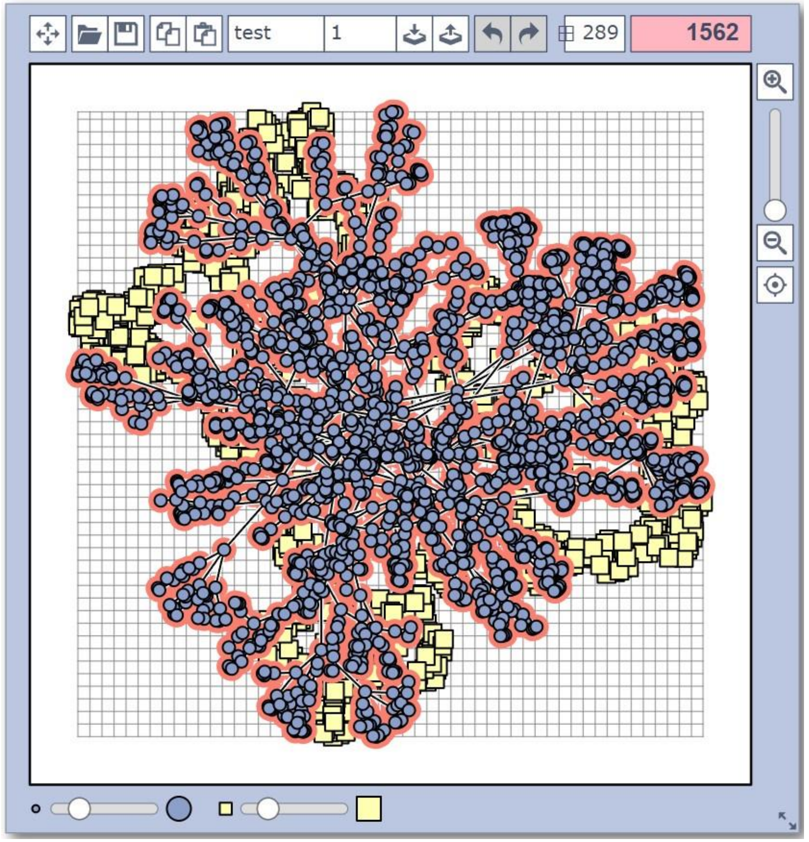
Approaches: FDAs – Eades vs. FR

Function	Eades	Fruchterman & Reingold
$f_{repl}(u,v)$	$c_{repl} \cdot \frac{1}{\|\|\overrightarrow{uv}\|\|^2} \cdot \overrightarrow{e_{vu}}$	$\frac{l^2}{\|\|\overrightarrow{uv}\|\|} \cdot \overrightarrow{e_{vu}}$
$f_{attr}(u,v)$	$c_{attr} \cdot \log \frac{\|\|\overrightarrow{uv}\|\|}{l} \cdot \overrightarrow{e_{uv}} - f_{repl}(u,v)$	$\frac{\|\|\overrightarrow{uv}\|\|^2}{l} \cdot \overrightarrow{e_{uv}}$
$\delta_t(\vec{v})$	$c^{(t)} \cdot \vec{v}$	$\begin{cases} m_t \cdot \overrightarrow{e_v}, & 2 \cdot l \cdot c^{(t)} < \|\|\vec{v}\|\| \\ \vec{v}, & otherwise \end{cases}$

- Suggested parameter combinations
  - Eades: $T = 128,\ c_{attr} = 10,\ \ c_{repl} = 10000,\ l = 5,\ c = 0.992$
  - FR: $T = 128,\ l = 100,\ c = 0.992$



Input



Output

Approaches: Greedy assignments – Fast implementation

Input: Drawing Γ of a Graph $G(V, E)$, point-set P
Output: PSE $\Gamma \in \mathbb{T}_{G,P}$, which strongly resembles Γ

foreach v **in** V **do**
  $d_{min} \leftarrow \infty$
  $p_{min} \leftarrow \text{null}$
  foreach p **in** P **do**
    if $\Gamma(p)^{-1} = \emptyset$ **then**
      $d \leftarrow \|\Gamma(v) - p\|$
      if $d < d_{min}$ **then**
        $d_{min} \leftarrow d$
        $p_{min} \leftarrow p$
  $\Gamma(v) \leftarrow p_{min}$
return Γ

- Problem: Vertices could be assigned to points even though others are closer.

Approaches: Greedy assignments – Slow implementation

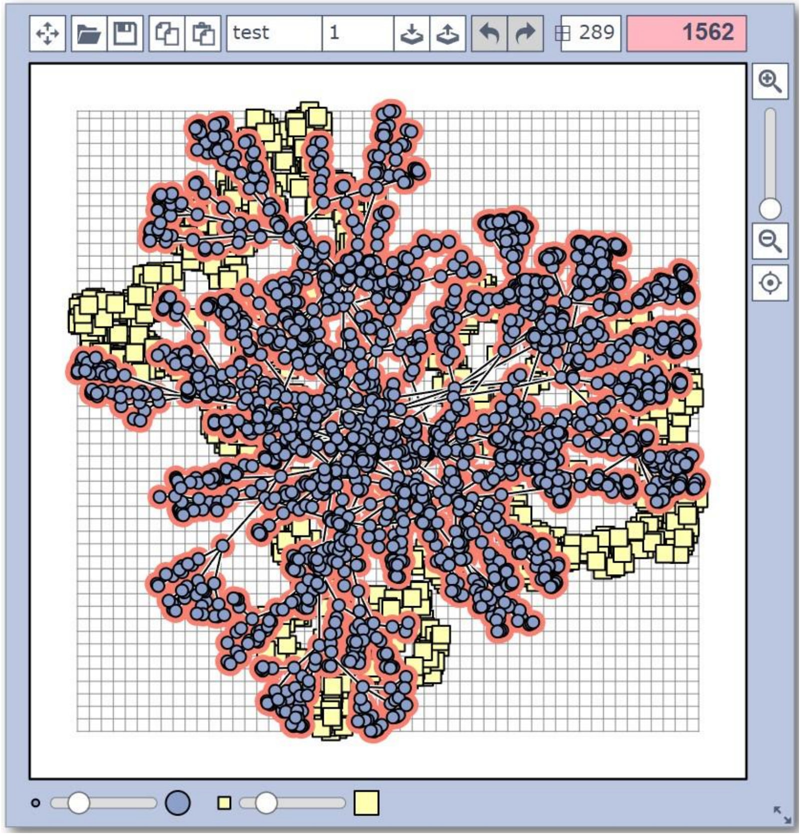
Input: Drawing Γ of a graph $G(V, E)$, point-set P
Output: PSE $\Gamma \in \mathbb{T}_{G,P}$, which strongly resembles Γ

while $\exists v \in V : \Gamma(v) \notin P$ **do**
    $d_{min} \leftarrow \infty$
    $p_{min} \leftarrow (\text{null}, \text{null})$
    foreach v **in** V **do**
        if $\Gamma(v) \notin P$ **then**
            foreach p **in** P **do**
                if $\Gamma(p)^{-1} = \emptyset$ **then**
                    $d \leftarrow \|\Gamma(v) - p\|$
                    if $d < d_{min}$ **then**
                        $d_{min} \leftarrow d$
                        $p_{min} \leftarrow (v, p)$
    $\Gamma(p_{min_1}) \leftarrow p_{min_2}$
return Γ

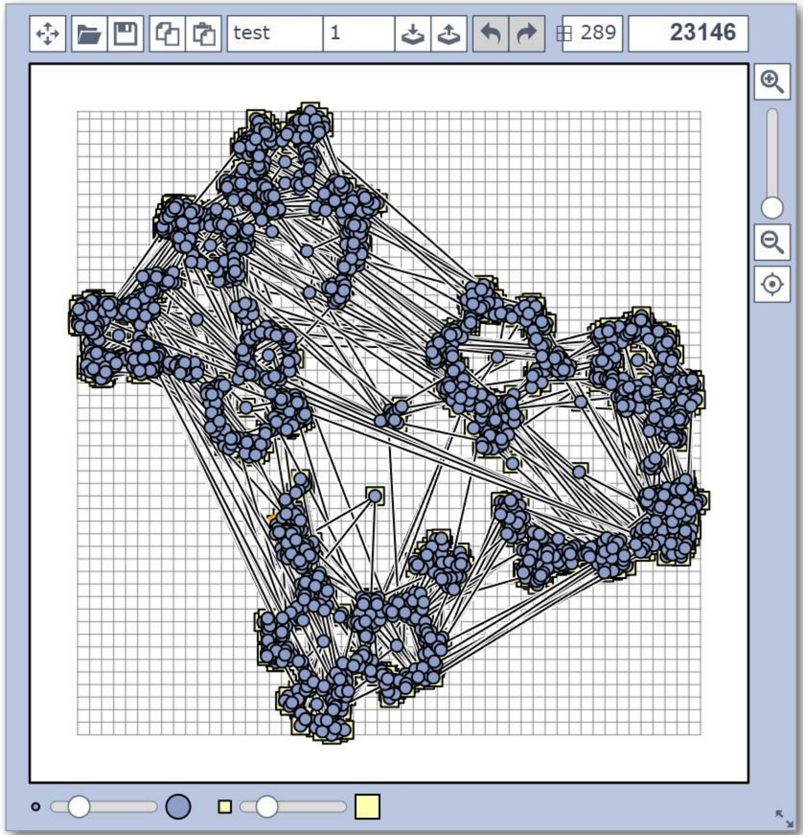
Approaches: Greedy assignments

FDA	Assignment	manual-1	manual-2	manual-3	manual-4	manual-5	manual-6	manual-7
Eades	Fast	14	**52**	670	53	**859**	1.572	320
Eades	Slow	14	69	600	37	1.453	1.422	**204**
FR	Fast	12	55	284	23	946	1.383	331
FR	Slow	**9**	69	**209**	**18**	864	**703**	332

FDA	Assignment	automatic-1	automatic-2	automatic-3	automatic-4	automatic-5	automatic-6	automatic-7
Eades	Fast	8.554.575	401.786	49.609.340	**10.445.229**	49.495.213	275.714	8.951.028.000
Eades	Slow	7.837.526	139.061	28.565.914	118.313.253	98.626.741	181.507	8.572.500.000
FR	Fast	**5.580.218**	342.506	16.688.138	18.165.757	32.690.218	**31.023**	3.218.733.000
FR	Slow	5.634.345	**106.037**	**7.670.227**	89.568.674	**31.572.197**	33.983	**2.600.058.000**



Input



Output

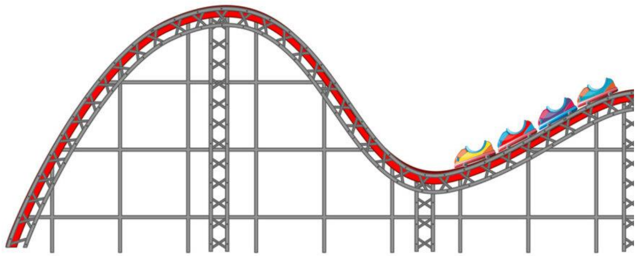
Approaches: SA

- Common global optimization strategy, which is capable of resolving local minima.

- “[...] *because of its analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration (i.e., its minimum lattice energy state), and thus is free of crystal defects.*”[3]

- Energy state ⟷ Score of the drawing
  - High temperature ⟶ Greater acceptance of deteriorations
  - Low temperature ⟶ Smaller acceptance of deteriorations

- <u>Acceptance criterion</u> by Metropolis: $A_{temp}(x) = \begin{cases} \exp(\frac{-x}{temp}), & x \geq 0 \\ 1, & x < 0 \end{cases}$



Approaches: SA

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$
Output: Variant of Γ with minimized crossings.

$\Gamma_{min} \leftarrow \Gamma$
while time remains **do**
  $\Gamma \leftarrow \Gamma_{min}$
  while less than s seconds have passed **do**
    $\Gamma' \leftarrow \text{Refactor}(\Gamma)$
    if $\text{Score}(\Gamma') < \text{Score}(\Gamma_{min})$ **then**
      $\Gamma_{min} \leftarrow \Gamma'$
    if random $x \in [0, 1) < A_{\delta_t(\theta)}(\text{Score}(\Gamma') - \text{Score}(\Gamma))$ **then**
      $\Gamma \leftarrow \Gamma'$
return Γ_{min}

- Exponential cooling: $\delta_t(\theta) = c^{(t)} \cdot \theta$
- Coolings suggested by Kutscheid:
  - Moderate: $\theta = 1,\ c = 0.94$
  - Rapid: $\theta = 2000,\ c = 0.85$
- Assessed frequencies: 5 seconds.

Approaches: SA – Random Walk

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$
Output: Refactored variant of Γ.

$p \leftarrow$ random $p \in P$
$v \leftarrow$ random $v \in V$
if $\Gamma(p)^{-1} = \emptyset$ **then**
  $\Gamma(v) \leftarrow p$
else
  $v' \leftarrow \Gamma(p)^{-1}$
  $\Gamma(v') \leftarrow \Gamma(v)$
  $\Gamma(v) \leftarrow p$
return Γ

- Dynamic probabilities for the randomized selection of vertices to be moved:

$$d_p(v) = \frac{pen(v)^p}{\sum_{u \in V} pen(u)^p}$$

  - $p = 0 \longrightarrow$ equally distributed probabilities
  - $p = 1 \longrightarrow$ linear consideration of penalties
  - $p = 2 \longrightarrow$ quadratic consideration of penalties

Approaches: SA – Rebuild Neighbourhood

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$

Output: Refactored variant of Γ.

$v \leftarrow$ random $v \in V$

$move \leftarrow N(v) \cup \{v\}$

$near \leftarrow \{|N(v)|\text{-nearest points to } \Gamma(v)\} \cup \{\Gamma(v)\}$

Shuffle($near$)

for $i = 0$ **to** $|near|$ **do**

    $v \leftarrow move[i]$

    $p \leftarrow near[i]$

    if random $x \in [0, 1] < \kappa$ **then**

        $p \leftarrow$ random $p \in P$

    MoveOrSwap(v, p)

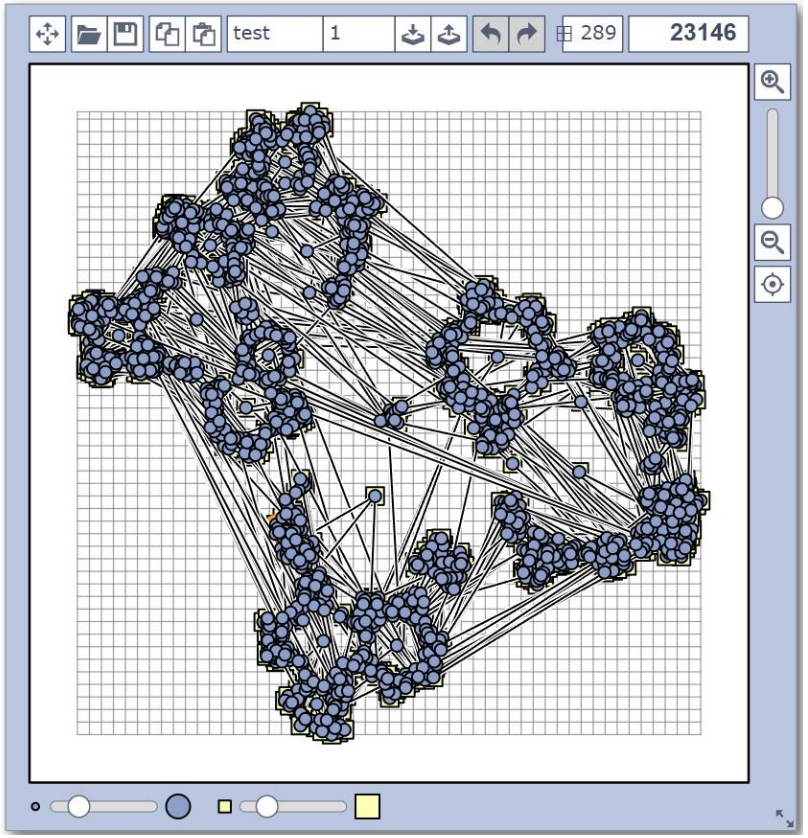
return Γ

- $\kappa \in [0,1]$ regulates the probability, with which distant points are also taken into account.
- Relatively small differences between a vertex selection with $p = 0$ and $p = 1$.

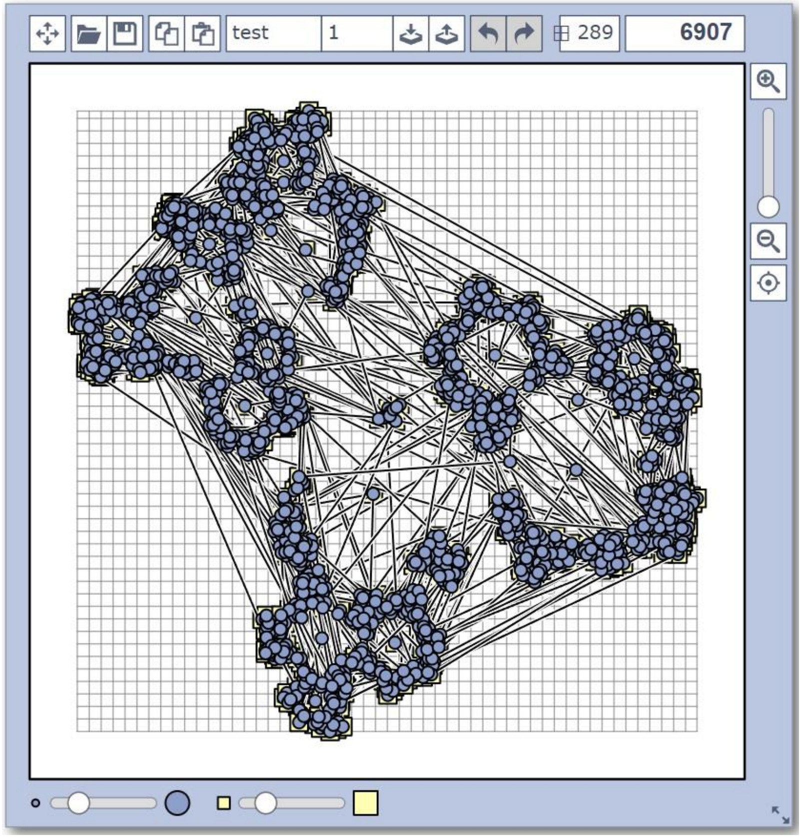
Approaches: SA

Refactoring	Cooling	p	manual-1	manual-2	manual-3	manual-4	manual-5	manual-6	manual-7
Random Walk	Moderate	1	3	25	0	4	4	**395**	19
Random Walk	Rapid	1	3	25	0	4	4	411	19
Rebuild Neighourhood	Rapid	0	3	25	1	5	8	537	25

Refactoring	Cooling	p	automatic-1	automatic-2	automatic-3	automatic-4	automatic-5	automatic-6	automatic-7
Random Walk	Moderate	1	**4.784.304**	64.101	1.088.192	2.182.424	**1.803.207**	14.977	1.138.653.000
Random Walk	Rapid	1	4.797.353	59.505	**618.957**	**2.174.089**	2.665.358	12.305	1.069.290.000
Rebuild Neighourhood	Rapid	0	5.580.218	**50.365**	2.316.571	8.932.152	7.976.241	**8.555**	**850.587.000**



Input



Output

Report GDC 2024

- Participated as team „Graph Gladiators“
  - Philipp Kindermann, Alexander Kutscheid, Jan-Niclas Loosen

Source	manual-1	manual-2	manual-3	manual-4	manual-5	manual-6	manual-7
Toolbox	11	0	2	8	84	24	15
Kindermann	11	0	2	-	12	-	-
GDC	11	0	2	8	12	24	15

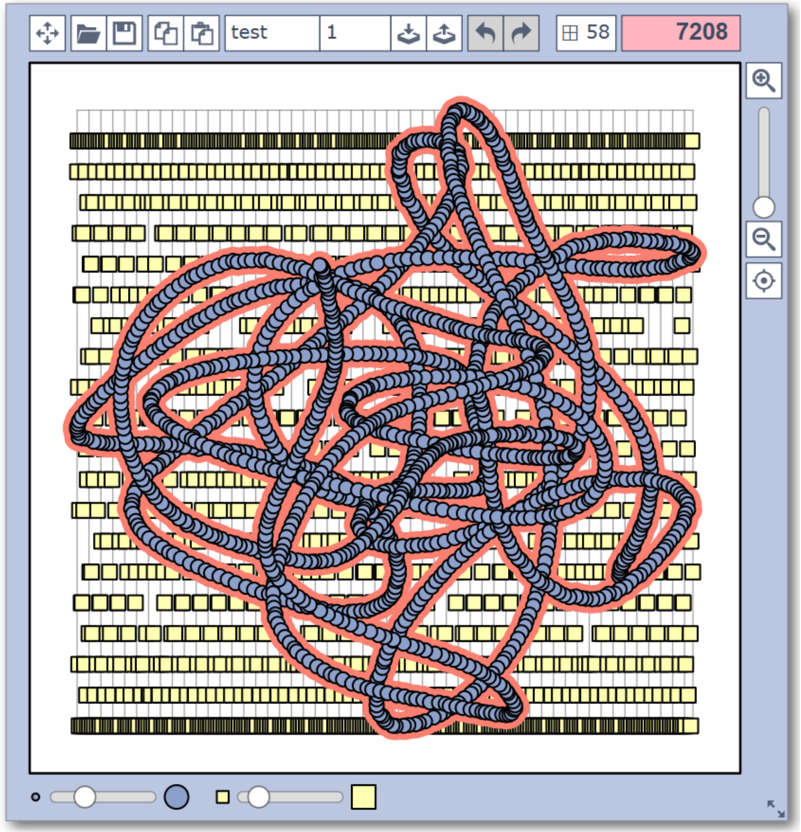
Source	automatic-1	automatic-2	automatic-3	automatic-4	automatic-5	automatic-6	automatic-7	automatic-8
Toolbox	192.211	307.724	38.588	6.277	3.705.859	908.423	1.536.688	12.619
Kutscheid	183.516	-	25.309	5.450	2.113.030	598.224	831.078	-
GDC	4.468	307.742	3.961	4	65.486	598.224	65.947	3.583

Conclusion

- Combined approach remains a promising methodology.
  - Even under competitive conditions.

- FR was clearly superior to the Eades implementations of spring embedding.
- Simulated annealing further improved by...
  - an increased reset frequency of $s = 5$ seconds.
  - dynamically calculated probabilities for the selection of the vertices to be modified.
  - Rebuild Neighbourhood as refactoring technique (at least for supported drawings).

Discussion



Important references

[1] **Peter Eades**: A heuristic for graph drawing. *Congressus numerantium*, 42(11):149–160, 1984. https://www.cs.ubc.ca/~will/536E/papers/Eades1984.pdf, visited on 26.09.2024.

[2] **Thomas M. J. Fruchterman** and **Edward M. Reingold**: Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. https://doi.org/10.1002/spe.4380211102, visited on 26.09.2024.

[3] **Darrall Henderson**, **Sheldon H. Jacobson**, and **Alan W. Johnson**: The theory and practice of simulated annealing. In **Fred Glover** and **Gary A. Kochenberger** (editors): *Handbook of Metaheuristics*, chapter 10, pages 287–319. Springer, 2003. https://dx.doi.org/10.1007/0-306-48056-5_10, visited on 26.09.2024.

[4] **Alexander Kutscheid**: Minimizing crossings in point-set embeddings, 2024.

[5] **Nicholas Metropolis**, **Arianna W. Rosenbluth**, **Marshall N. Rosenbluth**, **Augusta H.**, and **Edward Teller**: Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. https://bayes.wustl.edu/Manual/EquationOfState.pdf, visited on 26.09.2024.

[6] **Yaghout Nourani** and **Bjarne Andresen**: A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31:8373–8385, 1998. https://www.fys.ku.dk/~andresen/BAhome/ownpapers/perm-annealSched.pdf, visited on 26.09.2024.

[7] **P. C. Schuur**: Classification of acceptance criteria for the simulated annealing algorithm, volume 8929 of *Memorandum COSOR*. Technische Universiteit Eindhoven, 1989. https://pure.tue.nl/ws/portalfiles/portal/2116564/338267.pdf, visited on 26.09.2024.

[8] **Luca Vismara**, **Giuseppe Di Battista**, **Ashim Garg**, **Giuseppe Liotta**, **Roberto Tamassia**, and **Francesco Vargiu**: Experimental studies on graph drawing algorithms. *Software: Practice and Experience*, 30(11):1235–1284, 2000. https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-024X%28200009%2930%3A11%3C1235%3A%3AAID-SPE339%3E3.0.CO%3B2-B, visited on 26.09.2024.