Algorithms for Graph Visualization

Two Heuristics

1. Lecture
Summer Semester 2016

(based on slides from Martin Nöllenburg and Robert Görke, KIT)
How will we do it?

Given a graph $G = (V, E)$, find a ,,nice“ drawing $\Gamma$, where
\begin{itemize}
  \item vertices are points and
  \item edges are straight lines
\end{itemize}
How will we do it?

Given a graph $G = (V, E)$, find a „nice“ drawing $\Gamma$, where

- vertices are points and
- edges are straight lines

Ideas? Criteria?
How will we do it?

Given a graph $G = (V, E)$, find a „nice“ drawing $\Gamma$, where

- vertices are points and
- edges are straight lines
- similar edge lengths
- greater distances on non-adjacent vertices

Ideas? Criteria?
How will we do it?

Given a graph \( G = (V, E) \), find a „nice“ drawing \( \Gamma \), where

\[ \begin{align*}
\Rightarrow & \text{ vertices are points and } \\
\Rightarrow & \text{ edges are straight lines }
\end{align*} \]

I Ideas? Criteria?

Physical Analogy: attractive vs. repulsive forces.
How will we do it?

Given a graph $G = (V, E)$, find a „nice“ drawing $\Gamma$, where
- vertices are points and
- edges are straight lines

Ideas? Criteria?

Physical Analogy: attractive vs. repulsive forces.

Case 1: vertices $u, v$ adjacent

Case 2: $u$ and $v$ non-adjacent
How will we do it?

Given a graph \( G = (V, E) \), find a „nice“ drawing \( \Gamma \), where
- vertices are points and
- edges are straight lines

Ideas? Criteria?

Physical Analogy: attractive vs. repulsive forces.

Case 1: vertices \( u, v \) adjacent

\[
\begin{array}{c}
\text{Case 2: } u \text{ and } v \text{ non-adjacent}
\end{array}
\]
How will we do it?

Given a graph $G = (V, E)$, find a „nice“ drawing $\Gamma$, where vertices are points and edges are straight lines.

Ideas? Criteria?

Physical Analogy: attractive vs. repulsive forces.

Case 1: vertices $u, v$ adjacent

Case 2: $u$ and $v$ non-adjacent
Part I – Force Based Methods
An iterative approach

SpringEmbedder\((G = (V, E), p = (p_v)_{v \in V}, \varepsilon > 0, K \in \mathbb{N})\)

\[ t \leftarrow 1 \]

\[ \text{while } t < K \text{ and } \max_{v \in V} \|F_v(t)\| > \varepsilon \text{ do} \]

\[ \quad \text{foreach } v \in V \text{ do} \]

\[ \quad \quad F_v(t) \leftarrow \sum_{u:uv \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:uv \in E} f_{\text{spring}}(p_u, p_v) \]

\[ \quad \text{foreach } v \in V \text{ do} \]

\[ \quad \quad p_v \leftarrow p_v + \delta \cdot F_v(t) \]

\[ t \leftarrow t + 1 \]

\[ \text{return } p \]
An iterative approach

SpringEmbedder($G = (V, E), p = (p_v)_{v \in V}, \varepsilon > 0, K \in \mathbb{N}$)

$t \leftarrow 1$

while $t < K$ and $\max_{v \in V} \|F_v(t)\| > \varepsilon$ do

  foreach $v \in V$ do
  
  $F_v(t) \leftarrow \sum_{u:uv \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:uv \in E} f_{\text{spring}}(p_u, p_v)$

  foreach $v \in V$ do
  
  $p_v \leftarrow p_v + \delta \cdot F_v(t)$

  $t \leftarrow t + 1$

return $p$
An iterative approach

SpringEmbedder\((G = (V, E), \ p = (p_v)_{v \in V}, \ \varepsilon > 0, \ K \in \mathbb{N})\)

\[ t \leftarrow 1 \]

\[ \text{while } t < K \ \text{and} \ \max_{v \in V} \left\| F_v(t) \right\| > \varepsilon \ \text{do} \]

\[ \quad \text{foreach } v \in V \ \text{do} \]

\[ \quad \quad F_v(t) \leftarrow \sum_{u:uv \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:uv \in E} f_{\text{spring}}(p_u, p_v) \]

\[ \quad \quad \text{foreach } v \in V \ \text{do} \]

\[ \quad \quad \quad p_v \leftarrow p_v + \delta \cdot F_v(t) \]

\[ \quad t \leftarrow t + 1 \]

\[ \text{return } p \]
Analysis

Advantages
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.

Timing

In each iteration, we calculate all values of

– $f_{rep}$ $O(\text{ })$ time
– $f_{spring}$ $O(\text{ })$ time
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.

Timing

In each iteration, we calculate all values of

\[- f_{\text{rep}} \quad O(|V|^2) \text{ time} \]
\[- f_{\text{spring}} \quad O(\quad ) \text{ time} \]
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.

Timing

In each iteration, we calculate all values of

– \( f_{\text{rep}} \) \( O(|V|^2) \) time
– \( f_{\text{spring}} \) \( O(|E|) \) time
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.

Timing

In each iteration, we calculate all values of

- $f_{\text{rep}}$ \(O(|V|^2)\) time
- $f_{\text{spring}}$ \(O(|E|)\) time
Analysis

Advantages

+ very easy
+ surprisingly good for not too large graphs

Disadvantages

– possibly unstable
– ends too soon / stuck in a local minimum.

Timing

In each iteration, we calculate all values of

- $f_{\text{rep}}$: $O(|V|^2)$ time
- $f_{\text{spring}}$: $O(|E|)$ time

- $F_v$ and $p_v$: $O(|V|^2)$ time
Force Directed Spring-Embedder by Eades (1984)

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \frac{p_v - p_u}{l} \]

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \frac{p_u - p_v}{\|p_u - p_v\|} \]
Force Directed Spring-Embedder by Eades (1984)

\[
f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \overrightarrow{p_u p_v}
\]

\[
f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \overrightarrow{p_v p_u}
\]
Force Directed Spring-Embedder by Eades (1984)

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \overrightarrow{p_u p_v} \]

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \overrightarrow{p_v p_u} \]
Force Directed Spring-Embedder by Eades (1984)

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \overrightarrow{p_u p_v} \]

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \overrightarrow{p_v p_u} \]
Force Directed Spring-Embedder by Eades (1984)

Distance

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{|p_v - p_u|}{l} \cdot \overrightarrow{p_u p_v} \]

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{|p_v - p_u|} \cdot \overrightarrow{p_u p_v} \]
Force Directed Spring-Embedder by Eades (1984)

- **Force**
- **Spring constant (e.g. 2.0)**
  \[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{||p_v - p_u||}{l} \cdot \frac{p_v p_u}{||p_v - p_u||} \]
- **Distance**
- **Unit-vector from** \( p_v \) **to** \( p_u \)

- **Repulsive force**
  \[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{||p_v - p_u||} \cdot \frac{p_u p_v}{||p_v - p_u||} \]

- **Attractive force**
  \[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{||p_v - p_u||} \cdot \frac{p_u p_v}{||p_v - p_u||} \]
Force Directed Spring-Embedder by Eades (1984)

Distance

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \frac{p_u p_v}{l} \]

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \frac{p_v p_u}{l} \]

Spring constant (e.g. 2.0)

Force

Unit-vector from \( p_v \) to \( p_u \)

pull \( v \) to \( u \)

push \( v \) away
Force Directed Spring-Embedder by Eades (1984)

Distance

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \overrightarrow{p_u p_v} \]

Spring constant (e.g. 2.0)

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \overrightarrow{p_v p_u} \]

Unit-vector from \( p_v \) to \( p_u \)

Pull \( v \) to \( u \)

Push \( v \) away

Force
Force Directed Spring-Embedder by Eades (1984)

Distance

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \frac{p_u p_v}{\|p_u p_v\|} \]

Spring constant (e.g. 2.0)

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \frac{p_v - p_u}{\|p_v - p_u\|} \]

Unit-vector from \( p_v \) to \( p_u \)

Force

Pull \( v \) to \( u \)

Push \( v \) away
Force Directed Spring-Embedder by Eades (1984)

\[ f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\| p_v - p_u \|}{l} \cdot \frac{p_v - p_u}{\| p_v - p_u \|} \]

\[ f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\| p_v - p_u \|} \cdot \frac{p_u - p_v}{\| p_u - p_v \|} \]

Spring constant (e.g. 2.0)

Repulsion constant (e.g. 1.0)

Distance

Unit-vector from \( p_v \) to \( p_u \)
Force Directed Spring-Embedder by Eades (1984)

- **Distance**
  
  \[
  f_{\text{rep}}(p_u, p_v) = \frac{c_{\text{rep}}}{\|p_v - p_u\|} \cdot \frac{p_v - p_u}{\|p_v - p_u\|}
  \]

  - **Repulsion constant** (e.g. 1.0)
  - **Push** \(v\) away
  - **Pull** \(v\) to \(u\)

- **Force**
  
  \[
  f_{\text{spring}}(p_u, p_v) = c_{\text{spring}} \log \frac{\|p_v - p_u\|}{l} \cdot \frac{p_v - p_u}{\|p_v - p_u\|}
  \]

  - **Spring constant** (e.g. 2.0)
  - **Distance**
  - **Spring** constant
  - **Unit-vector from** \(p_v\) to \(p_u\)

  - **Natural** spring length

---

Steven Chaplick  
Lehrstuhl für Informatik I  
Universität Würzburg
Forces of Fruchterman & Reingold (1991)

\[ f_{\text{attr}}(p_u, p_v) = \frac{1}{\|p_u - p_v\|^2} \cdot \overrightarrow{p_v p_u} \]

\[ f_{\text{rep}}(p_u, p_v) = \frac{l^2}{\|p_u - p_v\|} \cdot \overrightarrow{p_u p_v} \]

\[ f_{\text{spring}} = f_{\text{rep}} + f_{\text{attr}} \]
Forces of Fruchterman & Reingold (1991)

\[ f_{\text{attr}}(p_u, p_v) = \frac{l^2}{\|p_u - p_v\|^2} \cdot \overrightarrow{p_v p_u} \]

\[ f_{\text{spring}} = f_{\text{rep}} + f_{\text{attr}} \]

\[ f_{\text{rep}}(p_u, p_v) = \frac{l^2}{\|p_u - p_v\|} \cdot \overrightarrow{p_u p_v} \]

for edges \( uv \)

for all vertex pairs \( u, v \)
Speeding up “convergence” – Via Grids

[Fruchterman & Reingold (1991)]
Speeding up “convergence” – Via Grids

[Fruchterman & Reingold (1991)]
Speeding up “convergence” – Via Grids

[Fruchterman & Reingold (1991)]
Speeding up “convergence” – Via Grids

[Fruchterman & Reingold (1991)]
Speeding up “convergence” – Via Grids

[Fruchterman & Reingold (1991)]
Speeding up – by adaptive displacement $\delta_v(t)$

Reminder...

SpringEmbedder\((G = (V, E), p = (p_v)_{v \in V}, \varepsilon > 0, K \in \mathbb{N})\)
\[ t \leftarrow 1 \]
\[ \textbf{while} \ t < K \ \textbf{and} \ \max_{v \in V} \|F_v(t)\| > \varepsilon \ \textbf{do} \]
\[ \textbf{foreach} \ v \in V \ \textbf{do} \]
\[ F_v(t) \leftarrow \sum_{u:uv \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:uv \in E} f_{\text{spring}}(p_u, p_v) \]
\[ \textbf{foreach} \ v \in V \ \textbf{do} \]
\[ p_v \leftarrow p_v + \delta \cdot F_v(t) \]
\[ t \leftarrow t + 1 \]
\[ \textbf{return} \ p \]
Speeding up – by adaptive displacement $\delta_v(t)$

Reminder...

SpringEmbedder($G = (V, E), p = (p_v)_{v \in V}, \varepsilon > 0, K \in \mathbb{N}$)

$t \leftarrow 1$

while $t < K$ and $\max_{v \in V} \|F_v(t)\| > \varepsilon$ do

    foreach $v \in V$ do
        $F_v(t) \leftarrow \sum_{u:uv \notin E} f_{\text{rep}}(p_u, p_v) + \sum_{u:uv \in E} f_{\text{spring}}(p_u, p_v)$

    foreach $v \in V$ do
        $p_v \leftarrow p_v + \delta \cdot F_v(t)$

    $t \leftarrow t + 1$

return $p$
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)

$$F_v(t) \Rightarrow \text{larger } \delta_v(t)$$
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)

$\Rightarrow$ smaller $\delta_v(t)$
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)
Speeding up – by adaptive displacement $\delta_v(t)$

(Frick, Ludwig, Mehldau 1995)

\[ F_v(t - 1) \]
\[ F_v(t) \]
\[ F_v'(t) \]
\[ \alpha_v(t) \]

$\Rightarrow$ smaller $\delta_v(t)$
Quad-Tree

\[ R_0 \]

\[ QT \]
Quad-Tree
Quad-Tree
Quad-Tree
Quad-Tree
Calculating repulsive forces (Barnes-Hut 1986)
Calculating repulsive forces (Barnes-Hut 1986)
Calculating repulsive forces (Barnes-Hut 1986)
Calculating repulsive forces (Barnes-Hut 1986)

\[ f_{\text{rep}}(R_i, p_u) := |R_i| \cdot f_{\text{rep}}(\sigma R_i, p_u) \]
Calculating repulsive forces (Barnes-Hut 1986)

\[ f_{\text{rep}}(R_i, p_u) := |R_i| \cdot f_{\text{rep}}(\sigma_{R_i}, p_u) \]

\[ f_{\text{rep}}(p_u) := \sum_{R_i \in \mathcal{R}_u} f_{\text{rep}}(R_i, p_u) \]

where \( \mathcal{R}_u \) = all children of nodes on the \( R_0-u \)-path in \( QT \)
Calculating repulsive forces (Barnes-Hut 1986)

\[ f_{\text{rep}}(R_i, p_u) := |R_i| \cdot f_{\text{rep}}(\sigma R_i, p_u) \]

\[ f_{\text{rep}}(p_u) := \sum_{R_i \in \mathcal{R}_u} f_{\text{rep}}(R_i, p_u) \]

where $\mathcal{R}_u = \text{all children of nodes on the } R_0-u\text{-path in } QT$
Calculating repulsive forces (Barnes-Hut 1986)

\[ f_{\text{rep}}(R_i, p_u) := |R_i| \cdot f_{\text{rep}}(\sigma_{R_i}, p_u) \]

\[ f_{\text{rep}}(p_u) := \sum_{R_i \in \mathcal{R}_u} f_{\text{rep}}(R_i, p_u) \]

where \( \mathcal{R}_u \) = all children of nodes on the \( R_0-u \)-path in \( QT \)
Part II – Multidimensional Scaling
Idea: measure and “match” the dissimilarity

Let $V = \{1, \ldots, n\}$ be a lot of objects.

Let $D \in \mathbb{R}^{n \times n}$ Matrix, where $d_{ij} \sim$ dissimilarity between obj. $i, j$. 
Idea: measure and “match” the dissimilarity

Let $V = \{1, \ldots, n\}$ be a lot of objects.

Let $D \in \mathbb{R}^{n \times n}$ Matrix, where $d_{ij} \sim$ dissimilarity between obj. $i, j$.

Goal: find $x_1, \ldots, x_n \in \mathbb{R}^2$, so that
Idea: measure and “match” the dissimilarity

Let $V = \{1, \ldots, n\}$ be a lot of objects.

Let $D \in \mathbb{R}^{n \times n}$ Matrix, where $d_{ij} \sim$ dissimilarity between obj. $i, j$.

**Goal:** find $x_1, \ldots, x_n \in \mathbb{R}^2$, so that

$$\left\| x_i - x_j \right\| \approx d_{ij} \quad \text{for all } i, j \in V$$
Idea: measure and “match” the dissimilarity

Let $V = \{1, \ldots, n\}$ be a lot of objects.

Let $D \in \mathbb{R}^{n \times n}$ Matrix, where $d_{ij} \sim$ dissimilarity between obj. $i, j$.

Goal: find $x_1, \ldots, x_n \in \mathbb{R}^2$, so that

$$\|x_i - x_j\| \approx d_{ij} \quad \text{for all } i, j \in V$$

For our drawing, how do we define the dissimilarity between two objects, i.e., two vertices?
Idea: measure and “match” the dissimilarity

Let $V = \{1, \ldots, n\}$ be a lot of objects.

Let $D \in \mathbb{R}^{n \times n}$ Matrix, where $d_{ij} \sim$ dissimilarity between obj. $i, j$.

**Goal:** find $x_1, \ldots, x_n \in \mathbb{R}^2$, so that

$$\|x_i - x_j\| \approx d_{ij} \quad \text{for all } i, j \in V$$

For our drawing, how do we define the dissimilarity between two objects, i.e., two vertices?

**Answer:** The *distance* between vertices, i.e., the length of a shortest path between them.
Classic Scaling

Let $D^{(2)} = (d_{ij}^2)_{1 \leq i,j \leq n}$ (note: not $D \times D$).
Classic Scaling

Let $D^{(2)} = (d_{ij}^2)_{1 \leq i, j \leq n}$ (note: not $D \times D$).

Compute Matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ of pseudo products.
Classic Scaling

Let $D^{(2)} = (d_{ij}^2)_{1 \leq i, j \leq n}$ (note: not $D \times D$).

Compute Matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ of pseudo products.

$B := -\frac{1}{2} J_n D^{(2)} J_n$, where $J_n = I_n - \frac{1}{n} (1_n 1_n^T)$, $1_n^T = (1, 1, \ldots, 1)$.

$I_n =$ identity matrix
Classic Scaling

Let $D^{(2)} = (d_{ij}^2)_{1 \leq i, j \leq n}$ (note: not $D \times D$).

Compute Matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ of pseudo products.

$B := -\frac{1}{2} J_n D^{(2)} J_n$, where $J_n = I_n - \frac{1}{n} (1_n 1_n^T)$, $1_n^T = (1, 1, \ldots, 1)

I_n = \text{identity matrix}$

Let $v_1, \ldots, v_n \in \mathbb{R}^n$ be the eigenvectors of the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R}$ of $B.$
Classic Scaling

Let \( D^{(2)} = (d_{ij}^2)_{1 \leq i, j \leq n} \) (note: not \( D \times D \)).

Compute Matrix \( B = (b_{ij}) \in \mathbb{R}^{n \times n} \) of pseudo products.

\[
B := -\frac{1}{2} J_n D^{(2)} J_n, \quad \text{where} \quad J_n = I_n - \frac{1}{n}(1_n 1_n^T), \quad 1_n^T = (1, 1, \ldots, 1)
\]

Let \( v_1, \ldots, v_n \in \mathbb{R}^n \) be the eigenvectors of the eigenvalues \( \lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R} \) of \( B \).

Let \( X = [x_1, \ldots, x_n]^T = [\sqrt{\lambda_1} v_1, \sqrt{\lambda_2} v_2] \).
Classic Scaling

Let $D^{(2)} = (d_{ij}^2)_{1 \leq i, j \leq n}$ (note: not $D \times D$).

Compute Matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ of pseudo products.

$B := -\frac{1}{2} J_n D^{(2)} J_n$, where $J_n = I_n - \frac{1}{n}(1_n 1_n^T)$, $1_n^T = (1, 1, \ldots, 1)$

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ be the eigenvectors of the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R}$ of $B$.

Let $X = [x_1, \ldots, x_n]^T = [\sqrt{\lambda_1} \mathbf{v}_1, \sqrt{\lambda_2} \mathbf{v}_2]$.

The minimum strain ($X$) = $\|B - XX^T\|_2 = \sum (b_{ij} - x_i^T x_j)^2$. 

I_n = identity matrix
Classic Scaling

Let \( D^{(2)} = (d_{ij}^2)_{1 \leq i,j \leq n} \) (note: not \( D \times D \)).

Compute Matrix \( B = (b_{ij}) \in \mathbb{R}^{n \times n} \) of pseudo products.

\[
B := -\frac{1}{2} J_n D^{(2)} J_n, \quad \text{where} \quad J_n = I_n - \frac{1}{n} (1_n 1_n^T), \quad 1_n^T = (1, 1, \ldots, 1)
\]

\( I_n \) = identity matrix

Let \( v_1, \ldots, v_n \in \mathbb{R}^n \) be the eigenvectors of the eigenvalues \( \lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R} \) of \( B \).

Let \( X = [x_1, \ldots, x_n]^T = [\sqrt{\lambda_1} v_1, \sqrt{\lambda_2} v_2] \).

The minimum strain \( (X) = \| B - XX^T \|_2 = \sum (b_{ij} - x_i^T x_j)^2 \).

Good: Find a solution with optimal strain.
Bad: numerical inconsistency, dimension reduction degeneracy.
Distance Scaling

Definition of strain: set $b_{ij} \rightarrow d_{ij}$ and $x_i^T x_j \rightarrow \|x_i - x_j\|$: 

$\text{(unweighted) stress}(X) = \sum_{i,j} (d_{ij} - \|x_i - x_j\|)^2$
Distance Scaling

Definition of strain: set $b_{ij} \rightarrow d_{ij}$ and $x_i^T x_j \rightarrow \|x_i - x_j\|$: 

$\text{(unweighted) stress}(X) = \sum_{i,j} (d_{ij} - \|x_i - x_j\|)^2$

$\text{(weighted) stress}(X) = \sum_{i,j} w_{ij}(d_{ij} - \|x_i - x_j\|)^2$
Distance Scaling

Definition of strain: set $b_{ij} \rightarrow d_{ij}$ and $x_i^T x_j \rightarrow \|x_i - x_j\|$: 

\[
\text{(unweighted) stress}(X) = \sum_{i,j} (d_{ij} - \|x_i - x_j\|)^2
\]

\[
\text{(weighted) stress}(X) = \sum_{i,j} w_{ij} (d_{ij} - \|x_i - x_j\|)^2
\]

where $w_{ij} \geq 0$, e.g., $w_{ij} = d_{ij}^q$. 
Distance Scaling

Definition of strain: set $b_{ij} \to d_{ij}$ and $x_i^T x_j \to \|x_i - x_j\|$: 

\[ \text{(unweighted) stress}(X) = \sum_{i,j} (d_{ij} - \|x_i - x_j\|)^2 \]

\[ \text{(weighted) stress}(X) = \sum_{i,j} w_{ij} (d_{ij} - \|x_i - x_j\|)^2 \]

where $w_{ij} \geq 0$, e.g., $w_{ij} = d_{ij}^q$. In graph drawing often $q = -2$. 
Examples: Classical and Distance Scaling

(a) classical scaling
(b) $q = 2$
(c) $q = 0$
(d) $q = -1$
(e) $q = -2$
(f) $q = -4$
Example (a) Classical Scaling
Example (e) Distance Scaling with $q = -2$