Minimizing Potential Energy in Constrained Physical Systems and Applications to Graph Drawing

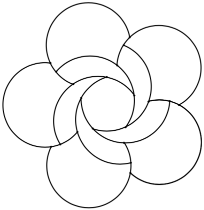
Bachelor Thesis of
Christian Schnorr

Agenda

- Introduction
  - Problem Statement
  - Motivation
  - Related Work
- Minimizing Potential Energy
- Drawing Graphs with Circular Arcs

Problem Statement

- Create drawings of arbitrary input graphs
  - Drawn edges as circular arcs
  - Use few geometric entities by drawing entire paths as a single circular arc
  - Visually appealing drawings

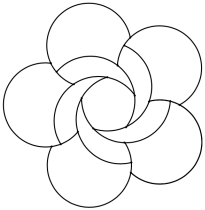


Motivation

- Drawings with few geometric entities are
  - easier to parse
  - easier to remember
- Drawings can be subject to any form of constraint
  - Soft constraints: violation decreases drawing's quality
  - Hard constraints: violation renders drawing invalid
  - Traditional force-directed algorithms may violate constraints

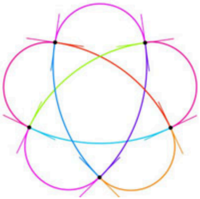
Related Work

- "Drawing Graphs with Few Arcs" (2015)
  - Lower and upper bounds of required entities
  - Only very specific classes of graphs



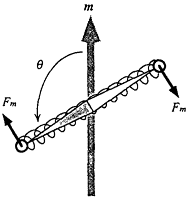
Related Work

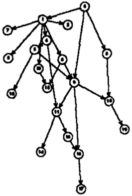
- "Force-directed Lombardi-style Graph Drawing" (2011)
  - Attempt to improve angular resolution
  - Soft constraint



Related Work

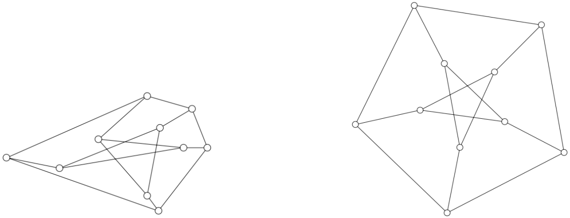
- Magnetic-spring algorithm (1994)
  - Attempt to align edges with a magnetic field
  - Soft constraint





Related Work

- PrEd (1999)
  - Preserves edge crossing properties
  - Hard constraint
  - Does not reduce the number of degrees of freedom



Agenda

- Introduction
- Minimizing Potential Energy
  - Generalized Coordinates
  - Forces and Potential Energy
  - Force-Directed Algorithms
  - Explicit Energy Function
- Drawing Graphs with Circular Arcs

Generalized Coordinates

- System of n particles with position vectors $\vec{r}_i \in \mathbb{R}^2$
  - Unconstrained systems: $2n$ degrees of freedom
  - Constrained systems: $\leq 2n$ degrees of freedom
- Constraints introduce dependencies between $\vec{r}_i$
  - Holonomic constraints $f_k(\vec{r}_1, \ldots, \vec{r}_n) \equiv 0$
  - Holonomic constraints reduce number of degrees of freedom
- Generalized coordinates q_j
  - Independent variables
  - Determine position vectors $\vec{r}_i = \vec{r}_i(q_1, \ldots, q_m)$
  - Implicitly satisfy holonomic constraints
  - $m =$ number of degrees of freedom

Forces and Potential Energy

$$W := \int -\vec{F}_{\text{res}}(\vec{r})\, d\vec{r} \tag{2.1}$$

- Restoring force $\vec{F}_{\text{res}}$ points towards equilibrium
- One must do work *against* $\vec{F}_{\text{res}}$ to displace system from equilibrium
- Potential energy $U = U_0 + W$

Forces and Potential Energy

$$\vec{F}_i := -\frac{\partial U}{\partial \vec{r}_i} \tag{2.2}$$

- Potential energy U is a function of position vectors $\vec{r}_i$
- $\vec{F}_i$ are directed towards lower energy levels
- Duality of potential energy and restoring forces

Force-Directed Algorithms

- Explicit restoring forces $\vec{F}_i$
- Potential energy U implicitly defined
- (Infinitesimal) displacements in direction of restoring forces reduce potential energy

Force-Directed Algorithms

- Restoring forces $\vec{F}_i$ act on particles
- Generalized forces Q_j
  - Translate restoring forces to generalized coordinates q_i

$$Q_j := \sum_{i=1}^{n} \vec{F}_i \cdot \frac{\partial \vec{r}_i}{\partial q_j}, \qquad j = 1, \ldots, m \tag{2.3}$$

$$\overset{(2.2)}{=} -\frac{\partial U}{\partial q_j} \tag{2.4}$$

- (Infinitesimal) displacements in the direction of generalized forces reduce potential energy

Explicit Energy Function

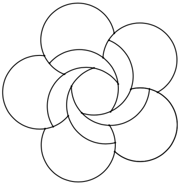
- Hard to find restoring forces that yield desired features in equilibrium
- Explicit potential energy $U\colon \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$
  - Easy to specify which features are desirable in equilibrium
  - Should be continuous and locally differentiable
- Generic optimization of U
  - Derivative-based optimization: e. g. Newton's method
  - Derivative-free optimization: e. g. Hill climbing

Agenda

- Introduction
- Minimizing Potential Energy
- Drawing Graphs with Circular Arcs
  - Definitions
  - Existence of Drawings
  - Graph Decomposition
  - Generalized Coordinates
  - Potential Energy
  - **??**
  - **??**

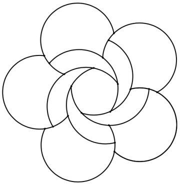
Definitions

- Drawing of G with circular arcs
  - Every edge $e \in E$ drawn as a circular arc Γ_e
  - No vertices coincide
  - No edges overlap
  - No edge intersects any vertices other than its endpoints



Definitions

- Given a set $\Pi = \{P_1, \ldots, P_k\}$ of edge-disjoint paths
  - $V(\Pi) \coloneqq V(P_1) \cup \ldots \cup V(P_k)$
  - $E(\Pi) \coloneqq E(P_1) \uplus \ldots \uplus E(P_k)$
- Drawing of Π with circular arcs
  - Drawing of $G \coloneqq (V(\Pi), E(\Pi))$ with circular arcs
  - Each path $P \in \Pi$ is drawn as a circular arc Γ_P



Existence of Drawings

- Circles are determined by three distinct points
- Necessary condition: $|V(P_i) \cap V(P_j)| \leq 2 \quad \forall i \neq j$
  - Condition is not sufficient!

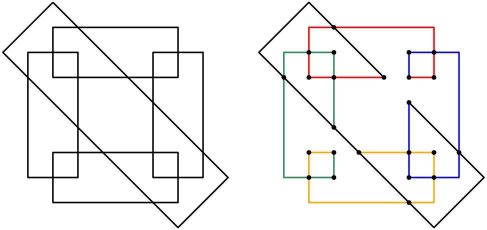


Figure: An arrangement of pseudocircles that is not circleable (left) and a derived arrangement of pseudo circular arcs (right).

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

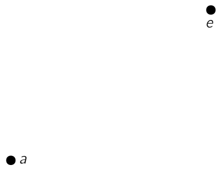


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

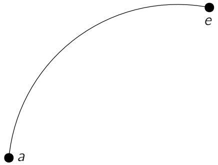


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

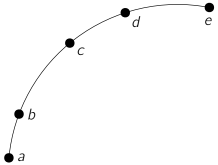


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

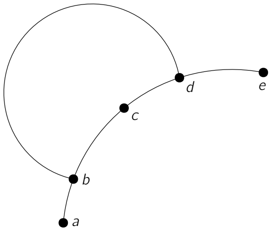


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

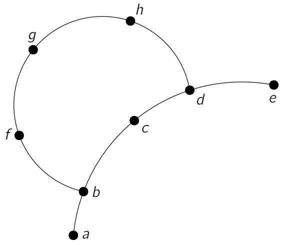


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

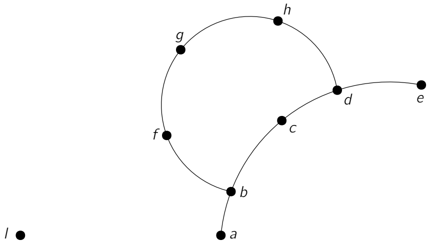


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

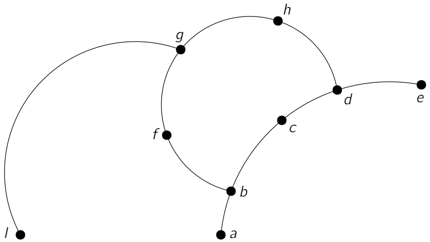


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

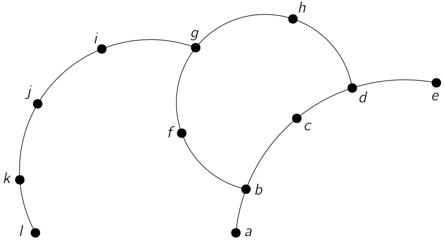


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm

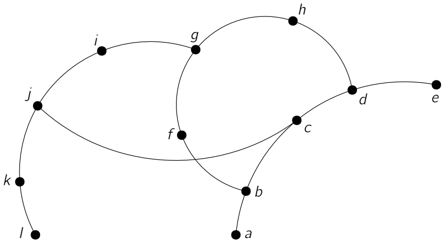


Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \dots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \dots \cup P_{i-1}) = \varnothing, \quad i = 2 \dots k$
  - Permits drawing with circular arcs using greedy algorithm

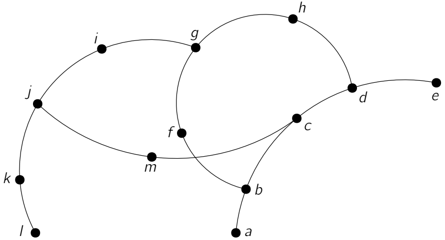


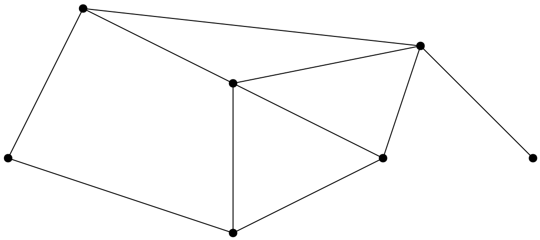
Figure: Greedy drawing of $\Pi = (abcde, bfghd, gijkl, jmc)$

Existence of Drawings

- Greedily realizable sequence of paths Π
  - Ordered sequence of edge-disjoint paths $\Pi = (P_1, \ldots, P_k)$
  - $V_{\text{int}}(P_i) \cap V(P_1 \cup \ldots \cup P_{i-1}) = \varnothing, \quad i = 2 \ldots k$
  - Permits drawing with circular arcs using greedy algorithm
- Characterization of vertices with respect to Π
  - Constrained vertices $V_{\text{c}}(\Pi)$: first appear as a path's internal vertex
  - Unconstrained vertices $V_{\text{u}}(\Pi)$: first appear as a path's endpoint

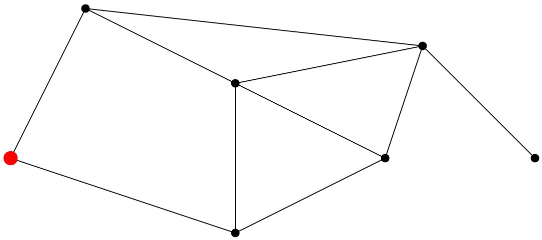
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



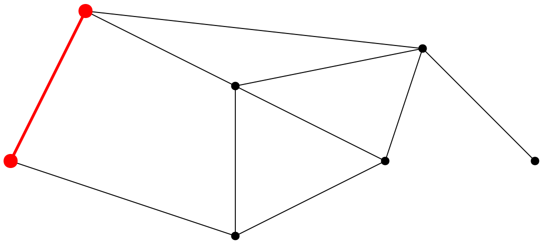
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



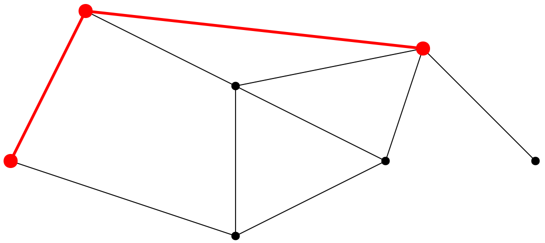
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



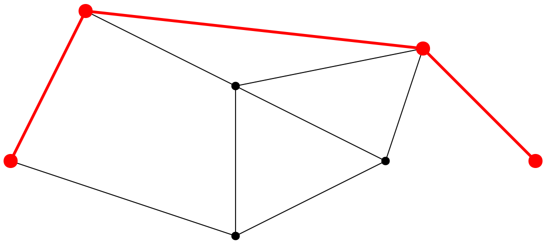
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



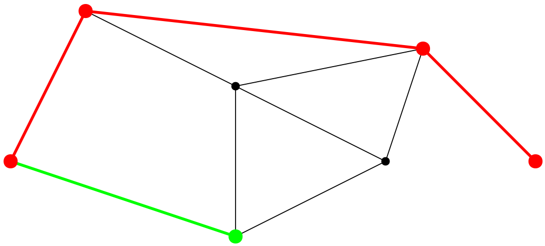
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



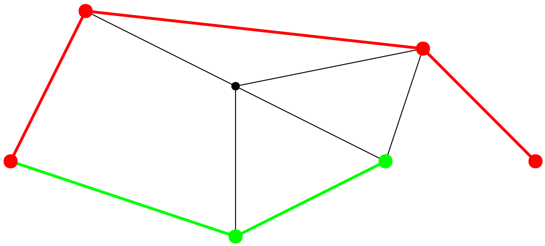
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



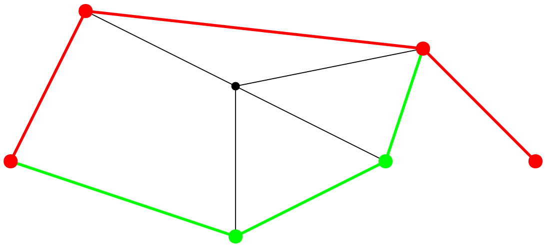
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



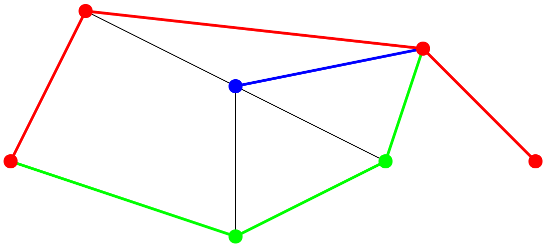
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



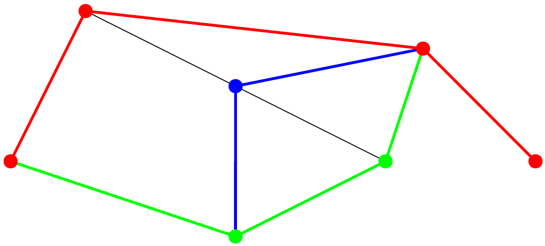
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



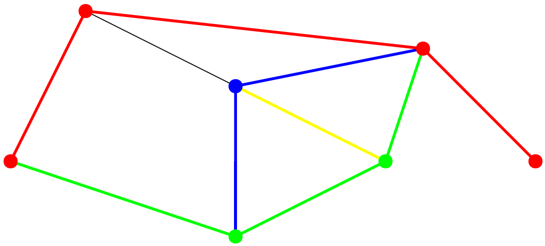
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



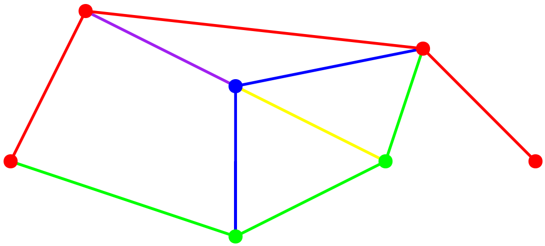
Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



Graph Decomposition

- Find decomposition into greedily realizable sequence of (simple) paths
- Trivial decomposition $\Pi = E$ valid
- Non-trivial (greedy) graph decomposition
  - Idea: append edges to working path until current head has been used before



Generalized Coordinates

- Generalized coordinates
  - $x\colon V_{\mathrm{u}}(\Pi) \to \mathbb{R}$
  - $y\colon V_{\mathrm{u}}(\Pi) \to \mathbb{R}$
  - $\varphi\colon \quad \Pi \to (-180^\circ, 180^\circ)$
  - $p\colon V_{\mathrm{c}}(\Pi) \to (0, 1)$

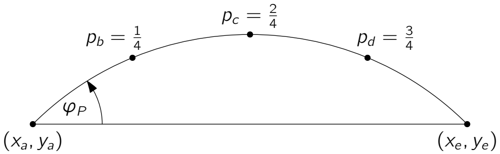


Figure: Generalized coordinates for a single path $P = abcde$ with equi-length edges.

Generalized Coordinates

foreach $v \in V_u(\Pi)$ **do**
    $\vec{r}(v) \leftarrow (x(v), y(v))$
end

foreach $P = (v_1, \ldots, v_n) \in \Pi$ **do**
    $\Gamma(P) \leftarrow \text{CircularArc}(\vec{r}(v_1), \vec{r}(v_n), \varphi(P))$

    foreach $v \in (v_2, \ldots, v_{n-1})$ **do**
        $\vec{r}(v) \leftarrow \Gamma(P).\text{pointForProgress}(p(v))$
    end
end

Algorithm 1: Transformation to Cartesian coordinates $\vec{r}(v)$

Potential Energy

- Vertex-Vertex repulsion
  - Repulsive force based on Coulomb's law
  - $F_{\text{rep}}(d) := c_1 \cdot \frac{1}{d^2}$

$$U_{\text{rep}}(u, v) := \int_{\infty}^{d(u,v)} -F_{\text{rep}}(s)\, ds$$

- Vertex-Vertex attraction
  - Attractive force based on logarithmic spring
  - $F_{\text{att}}(l) := -c_2 \cdot k \cdot \ln\left(\frac{l}{k}\right)$

$$U_{\text{att}}(e) := \int_{k}^{l(\Gamma_e)} -F_{\text{att}}(s)\, ds$$

Potential Energy

- Order of vertices on a path
  - Prevents intra-path overlapping edges

$$U_{\text{ord}}(P) := \begin{cases} 0 & \text{if edges } e \in E(P) \text{ do not overlap} \\ \infty & \text{otherwise} \end{cases}$$

- Vertex-Path repulsion
  - Prevents unwanted vertex-edge intersections
  - Prevents inter-path overlapping edges

$$U_{\text{int}}(v, P) := \begin{cases} c_3 \cdot \frac{1}{d(v, \Gamma_P)} & \text{if } d(v, \Gamma_P) \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

Potential Energy

$$U := \sum_{\{u,v\} \in V^2} U_{\text{rep}}(u, v) + \sum_{e \in E} U_{\text{att}}(e) + \sum_{P \in \Pi} U_{\text{ord}}(P) + \sum_{\substack{v \in V, P \in \Pi \\ v \notin V(P)}} U_{\text{int}}(v, P)$$

- Can be evaluated in $\mathcal{O}(|V|^2 + |E| + |V| \cdot |\Pi|)$
- Optimization using hill climbing

Results

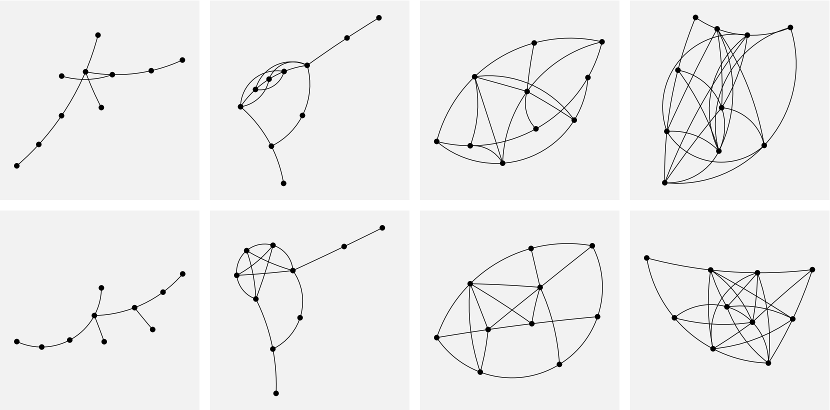


Figure: Drawings of graphs with 10 vertices and 9/15/20/25 edges; each with (bottom) and without (top) user adjustments.

Results

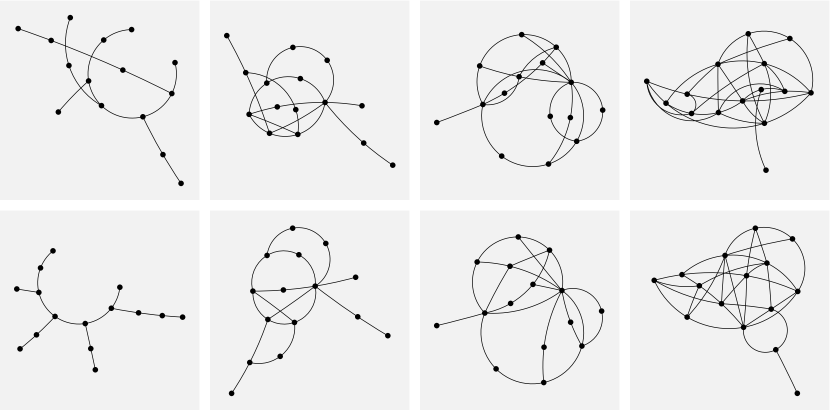


Figure: Drawings of graphs with 15 vertices and 14/20/25/35 edges; each with (bottom) and without (top) user adjustments.

Demo

- Implemented as a Mac OS Application
- Written in Swift 3.0