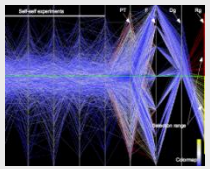
# Informations-visualisierung

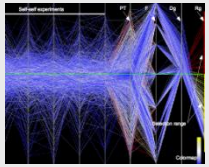
| | |
|---|---|
| Thema: | 5. Darstellung von Graphen |
| Dozent: | Prof. Dr. Gerik Scheuermann<br>scheuermann@informatik.uni-leipzig.de |
| Sprechstunde: | nach Vereinbarung |
| Umfang: | 2 |
| Prüfungsfach: | Modul Fortgeschrittene Computergraphik<br>Medizininformatik, Angewandte Informatik |



# Darstellung von Graphen

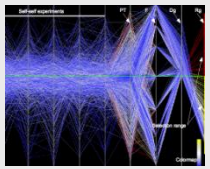
- Zu visualisierende Entitäten besitzen häufig **inhärente Relation** (Beziehung oder Abhängigkeit)
- Dann: **Graphen eignen sich als Repräsentation**, Knoten stellen Entitäten, Kanten die Relation dar
- Dieser Fall tritt häufig auf: **Verzeichnishierarchie** eines Computersystems
- Typische Fragestellungen:
  - Wo bin ich?
  - Wo ist die gesuchte Datei?

- Weitere typische Anwendungen
  - Organigramm einer Firma
  - Projektmanagement (z. B. Project Evaluation and Review Technique - PERT)
  - Soziale Netzwerke
  - Taxonomie biologischer Arten (bei Abstammung: phylogenetische Bäume)
  - Biochemische Reaktionen, speziell Stoffwechselsysteme oder Signalwege



# Darstellung von Graphen

- Ähnlichkeiten von Dokumenten
- Semantische Netzwerke und Wissensrepräsentationsdiagramme
- Webseiten und ihre Links
- Szenengraphen in der virtuellen Realität
- Datenstrukturen, insbesondere bei Compilern
- Strukturen objektorientierter Systeme (Klassenhierarchie, Datenfluss)
- Realzeitsystem (Zustandsdiagrame)
- Schaltkreisentwurf (Very Large System Integration VLSI)



# 5.1 Definition

## Graphen

Ein (einfacher) **Graph** G=(V,E) besteht aus endlicher Menge V von **Knoten** und endlicher Menge $E \subset \{\{v_i,v_j\} \wedge i \neq j \wedge v_i,v_j \in V\}$ **ungerichteter Kanten**.

Wenn {u,v} Kante ist, so heißen u und v **adjazent** (benachbart).

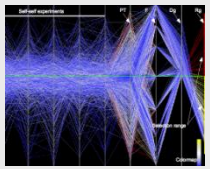
**Nachbarn** von $v \in V$ sind alle adjazenten Knoten.

Anzahl der Nachbarn ist **Grad** von v: $|v|$.

Ein **Weg** in Graph G ist Folge $(v_1,v_2,...,v_h)$ verschiedener Knoten von G, so dass $\{v_i,v_{i+1}\}$ Kante für i=1,...,h-1 ist.

Ein Weg heißt **Zyklus**, falls $\{v_h,v_1\}$ Kante ist.

Ein Graph ohne Zyklus heißt **azyklischer Graph.**





## Graphen (Fortsetzung)

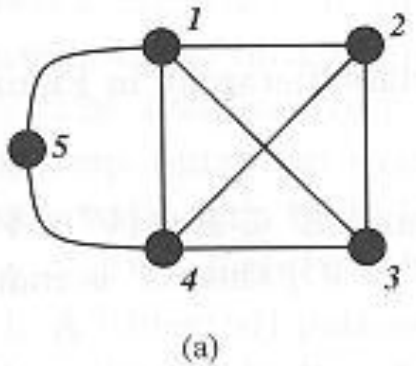
Ein Graph (V',E') mit $V' \subseteq V$ und $E' \subseteq E \cap (V' \times V')$ heißt **Teilgraph von G.**

Im Fall $E' = E \cap (V' \times V')$ heißt G' durch V' **induzierter Teilgraph von G.**

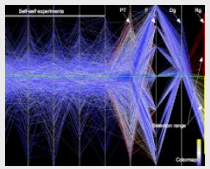
Graph G=(V,E) mit n Knoten kann durch n x n **Adjazenzmatrix** A beschrieben werden, wobei $A_{uv} = 1$ wenn $\{u,v\} \in E$, und $A_{uv}=0$ sonst.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | 1 | 0 |
| 3 | 1 | 1 | 0 | 1 | 0 |
| 4 | 1 | 1 | 1 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 | 0 |

**Table 1.1**: An adjacency matrix for graph $G_1$ shown in Figure 1.5.a.




(a)





## Digraphen

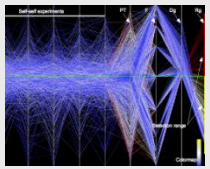
Ein gerichteter **Graph** oder **Digraph** G=(V,E) besteht aus endliche Menge V von **Knoten** und endliche Menge $E \cap V \times V$ **gerichteter Kanten.**

Die gerichtete Kante (u,v) ist **ausgehende Kante** für u, und **eingehende Kante** für v.

Knoten ohne ausgehende Kanten heißen **Senken**, Knoten ohne eingehende Kanten heißen **Quellen.**

**Ingrad** von v ist Anzahl eingehender Kanten.

**Ausgrad** von v ist Anzahl ausgehender Kanten.





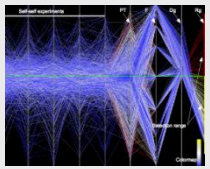
## Digraphen (Fortsetzung)

Ein **gerichteter Weg in G** ist Folge $(v_1,v_2,...,v_h)$ von Knoten in G, so dass $(v_i,v_{i+1})$ gerichtete Kante in G für i=1,...,h-1 ist.

Ein gerichteter Weg heißt **Zyklus**, falls $(v_h,v_1)$ gerichtete Kante in G ist.

Ein **gerichteter azyklischer Graph (DAG)** ist gerichteter Graph ohne Zyklus.

Zu einem Digraph gibt es immer Graph G, der durch „Vergessen der Orientierung der Kanten“ entsteht.

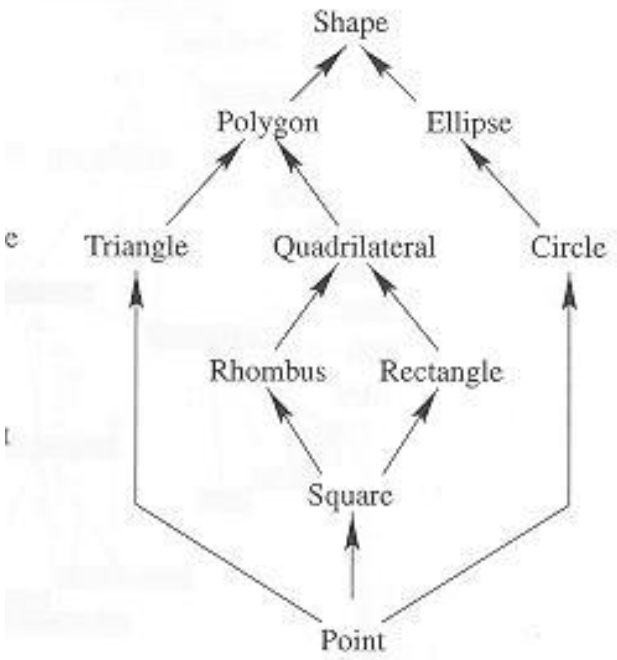




## Digraphen (Fortsetzung)

Kante (u,v) eines Digraphen G heißt **transitiv**, falls es einen Weg von u nach v gibt, der nicht durch (u,v) läuft.

**Transitiver Abschluss von G** ist Graph mit gleicher Knotenmenge und Kante (u,v) für jeden Weg von u nach v in G.



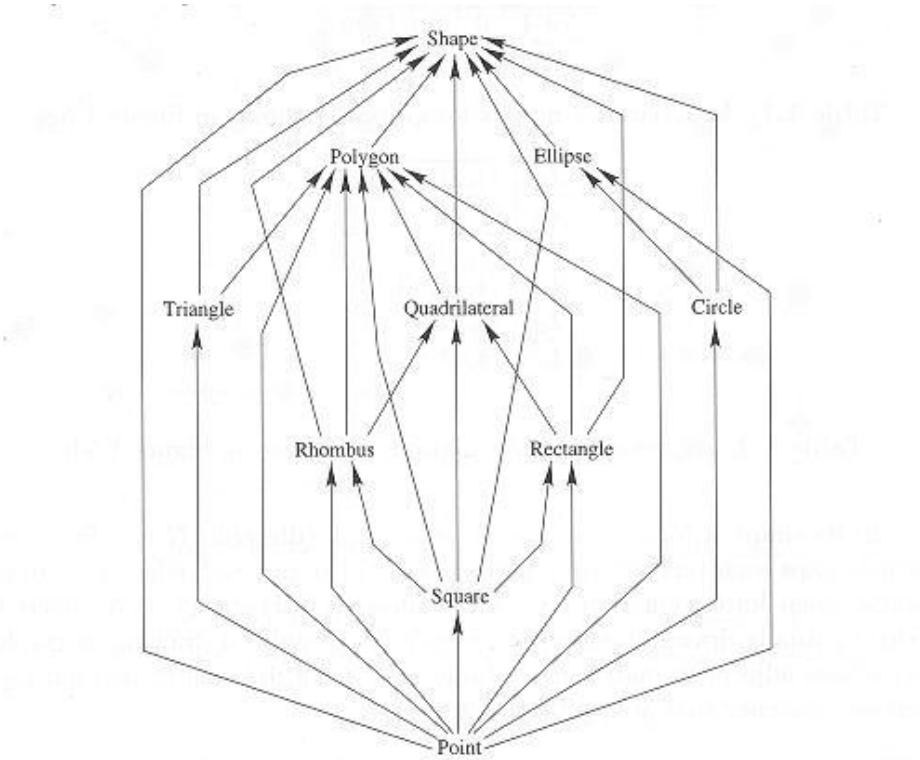




**Figure 1.4**: The transitive closure of the class hierarchy in Figure 1.3.

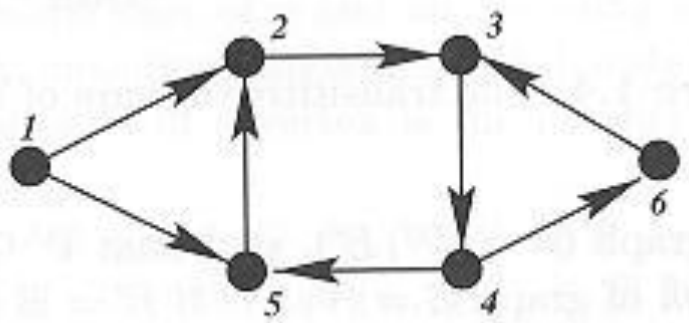
## Digraphen (Fortsetzung)

**Adjazenzmatrix** A **eines Digraphen** G=(V,E) mit n Knoten ist n x n Matrix mit $A_{uv}$=1 für (u,v) ∈ E und $A_{uv}$=0 sonst;. A ist im Allgemeinen **nicht symmetrisch.**

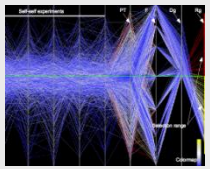
Adjazenzliste eines Digraphen ist **Liste**, mit einem Eintrag für jede Kante.

| | |
|---|---|
| $L_1$ | $(1,2), (1,5)$ |
| $L_2$ | $(2,3)$ |
| $L_3$ | $(3,4)$ |
| $L_4$ | $(4,5), (4,6)$ |
| $L_5$ | $(5,2)$ |
| $L_6$ | $(6,3)$ |

**Table 1.2**: Adjacency lists for digraph $G_2$ shown in Figure 1.5.b.




(b)



# 5.1 Definition

Eine **Zeichnung** eines Graphen oder Digraphen G ist Abbildung $\Gamma$ vom Graphen in Ebene:

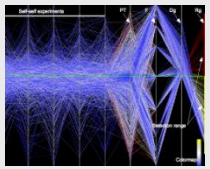
- Sie ordnet  jedem Knoten v einen Punkt G(v) und jeder Kante eine offene Jordankurve G(u,v) zu, die die Punkte G(u) und G(v) als Endpunkte hat.
- Gerichtete Kanten eines Digraphen werden in der Regel **als Pfeile** gezeichnet.

Eine **Zeichnung** eines Graphen oder Digraphen heißt **planar**, wenn sich Kurven zweier verschiedener Kanten höchstens in Endpunkten schneiden.

Ein **Graph oder Digraph heißt planar**, wenn es mindestens eine planare Zeichnung gibt.

Man beachte, dass **nicht alle Graphen planar** sind.

Nach der **Eulerformel** kann ein planarer Graph mit n Knoten **höchstens** 3n-6 Kanten haben!



# 5.1 Definition

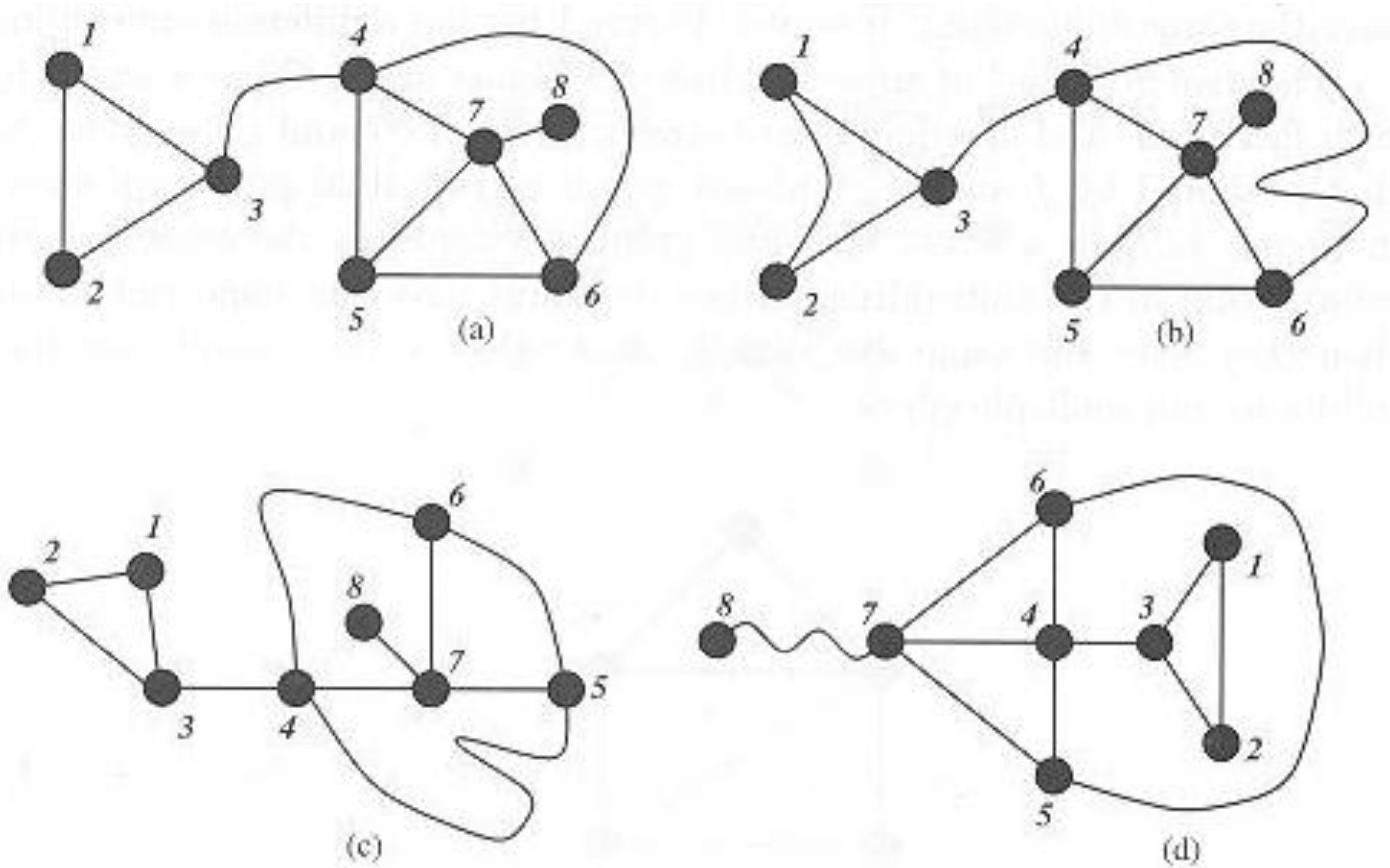
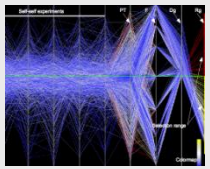



**Figure 1.6**: Four planar drawings of the same graph.



# 5.1 Definition

Eine planare Zeichnung eines Graphen oder Digraphen zerlegt die Ebene in topologisch verbundene Regionen. Diese werden **Facetten** genannt.

Die unbegrenzte Facette wird **äußere Facette** genannt.

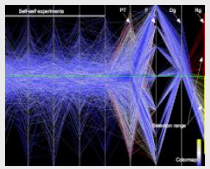
Eine planare Zeichnung bestimmt zirkuläre Ordnung der inzidenten Kanten an jedem Knoten **durch Umlauf gegen/im Uhrzeigersinn.**

Zwei planare Zeichnungen heißen **äquivalent**, wenn sie die **gleiche zirkuläre Ordnung** an allen Knoten festlegen.

Eine **planare Einbettung** eines Graphen oder Digraphen G ist **Äquivalenzklasse von planaren Zeichnungen:**

- Wird durch **Festlegen der zirkulären Ordnungen** der inzidenten Kanten an jedem Knoten festgelegt
- Auf der **vorherigen Folie** liefern a,b,c die gleiche, d dagegen eine andere Einbettung des Graphen.

Ein **eingebetteter Graph** / **Digraph** ist ein Graph / Digraph mit festgelegter planarer Einbettung.



# 5.1 Definition

Der **Dualgraph** G* zu einer Einbettung eines planaren Graphen G besitzt

- einen Knoten für jede **Facette der Einbettung** und
- eine Kante (f,g) zwischen zwei Facetten, wenn diese **Facetten eine gemeinsame Kante** in der Einbettung haben

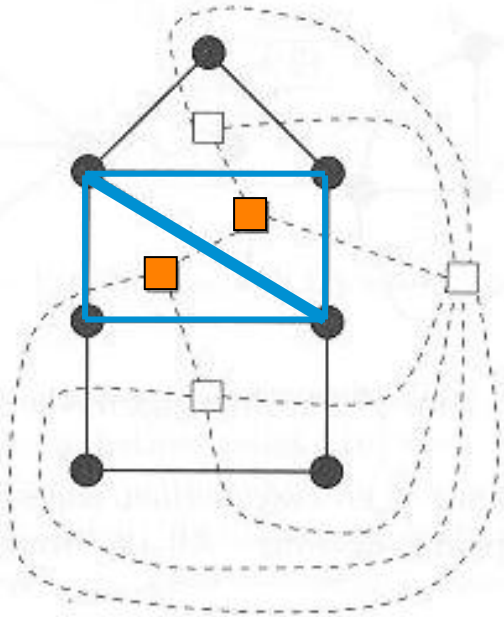
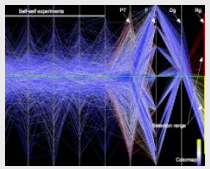


**Figure 1.7**: Dual graph (shown with boxes for vertices and dashed lines for edges) of an embedding of a planar graph.



# 5.1 Definition

Der **Dualgraph** G* zu einer Einbettung eines planaren Graphen G besitzt

- einen Knoten für jede **Facette der Einbettung** und
- eine Kante (f,g) zwischen zwei Facetten, wenn diese **Facetten eine gemeinsame Kante** in der Einbettung haben.

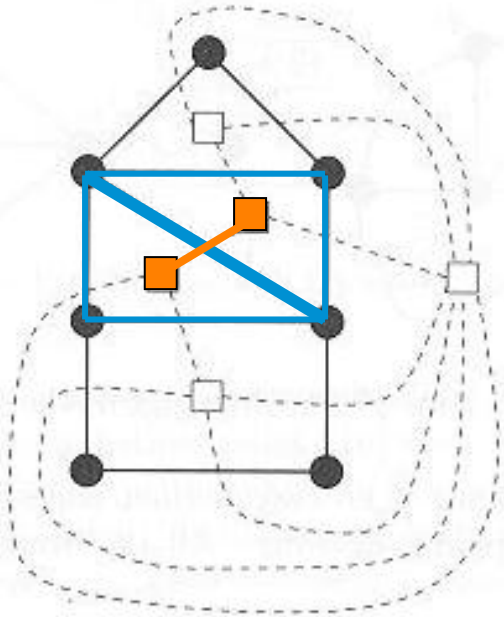
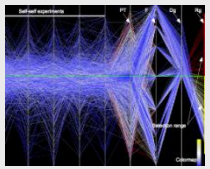


**Figure 1.7**: Dual graph (shown with boxes for vertices and dashed lines for edges) of an embedding of a planar graph.



# 5.1 Definition

Ein Graph heißt **zusammenhängend**, wenn es Weg zwischen beliebigen Knoten u und v gibt.

Ein maximaler zusammenhängender Teilgraph von G heißt **Zusammenhangskomponente.**

Ein **Schnittknoten** eines Graphen G ist ein Knoten, dessen Entfernen (einschließlich der inzidenten Kanten) den Graphen G in mehrere Zusammenhangskomponenten teilt.

Ein Graph ohne Schnittknoten heißt **doppelt zusammenhängend.**

Maximale doppelt zusammenhängende Teilgraphen heißen **Blöcke.**

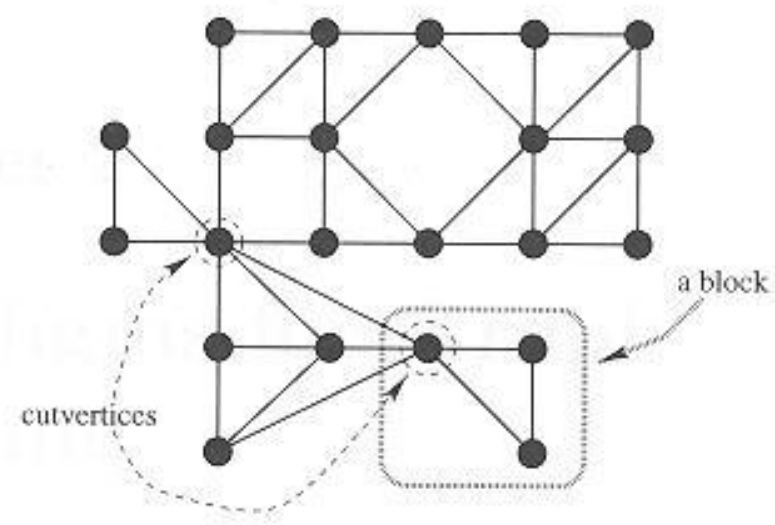
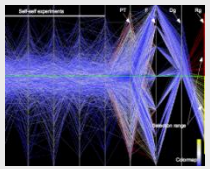



Figure 1.8: Cutvertices and blocks.



# 5.1 Definition

Ein Paar (u,v) von Knoten eines Graphen heißt **trennendes Paar**, falls Entfernen von u und v aus Graphen den Graphen teilt .

Ein doppelt zusammenhängender Graph ohne trennendes Paar heißt **dreifach verbunden** (**dreifach zusammenhängend**).

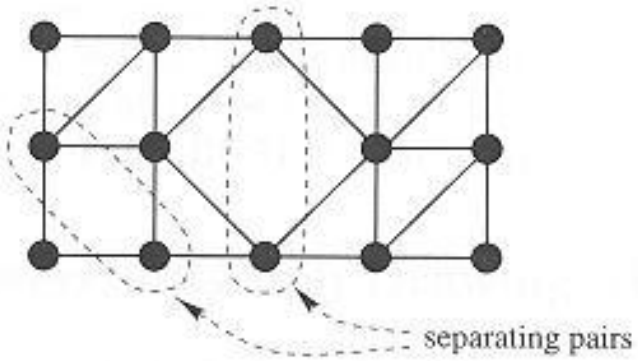
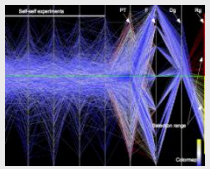



Figure 1.9: Separating pairs.

**Satz**: Skelett (Ecken und Kanten) eines Polyhedrons im Raum ist planarer, dreifach zusammenhängender Graph.

**Satz**: Planarer, dreifach zusammenhängender Graph hat eindeutige Einbettung (bis auf Umdrehung der zirkulären Ordnung an allen Knoten).



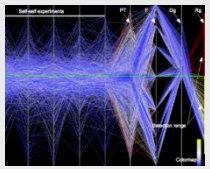


## Weitere Graphentypen

- **Multigraphen.** Sowohl bei **gerichteten als auch ungerichteten** Graphen können mehrfache Kanten zwischen gleichen Knoten und Kanten von einem Knoten zu sich selbst auftreten

- **Ungerichtete Hypergraphen**. Wenn eine Kante mehr als zwei Knoten verbinden kann, spricht man von Hypergraphen:
  Tupel (V,E) mit endlichen Menge von Knoten V und Menge $E \subseteq Pot(V)$ von Hyperkanten

- **Gerichtete Hypergraphen**. Ein Tupel (V,E) einer endlichen Menge V von Knoten und einer Menge E von Hyperkanten.
  **Jede Hyperkante verbindet** eine Ausgangsmenge von Knoten mit einer Eingangsmenge von Knoten.

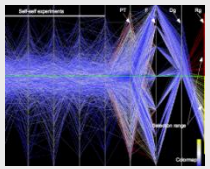




## Konventionen

Es gibt verschiedene **Zeichenkonventionen mit entscheidenden Einfluss** auf die Auslegealgorithmen von Graphen. Die gebräuchlichen Konventionen sind:

- **Polylinienzeichnung**. Jede Kante wird als Polygonzug gezeichnet.
- **Geradlinige Zeichnung.** Jede Kante wird als gerade Linie gezeichnet.
- **Orthogonale Zeichnung**: Jede Kante wird als Polygonzug achsenparalleler Linien gezeichnet.
- **Gitterzeichnung**. Knoten, Schnittpunkte und Kantenknicke haben ganzzahlige Koordinaten bzgl. eines festgelegten kartesischen Gitters.
- **Planare Zeichnung**. Kanten schneiden sich höchstens an Knoten.
- **Aufwärtszeichnung** / **Abwärtszeichnung.** Für azyklische gerichtete Graphen (DAGs) wird jede Kante als monoton ansteigende / absteigende Kurve gezeichnet.



# 5.2 Paradigmen

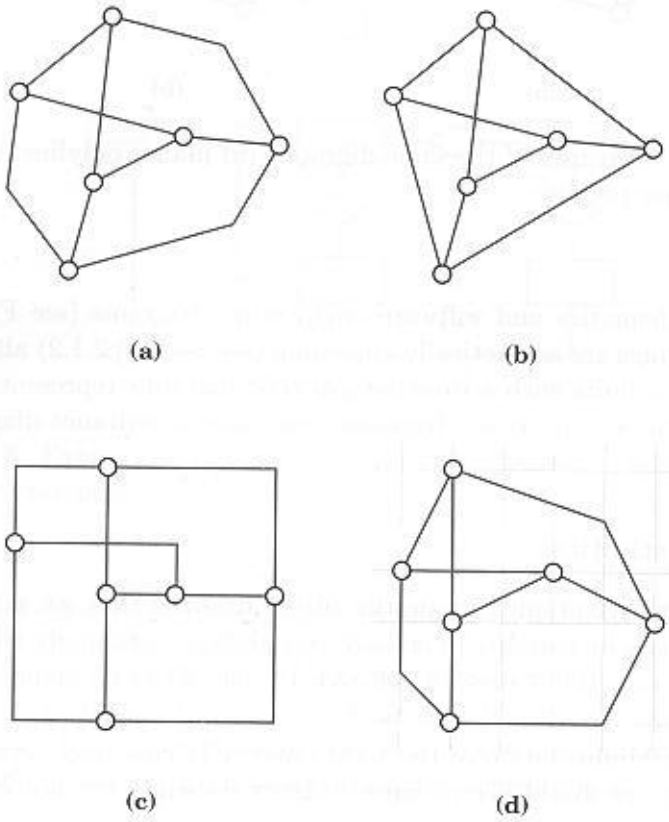



**Figure 2.1**: Drawings of the same graph: (a) polyline; (b) straight-line; (c) orthogonal; (d) polyline grid.

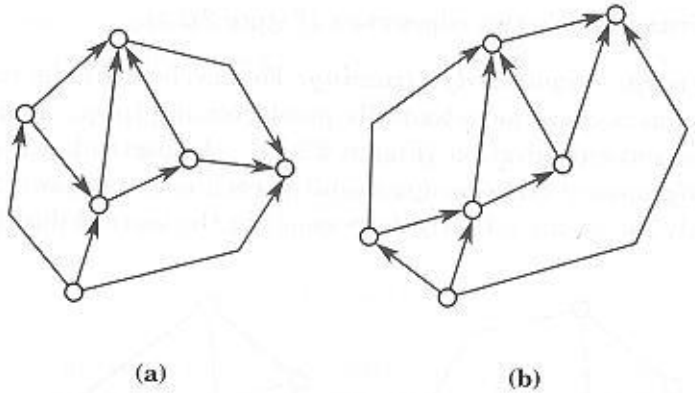
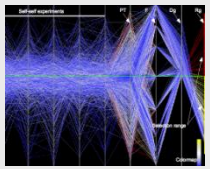



**Figure 2.2**: Drawings of the same digraph: (a) planar polyline; (b) strictly upward planar polyline.



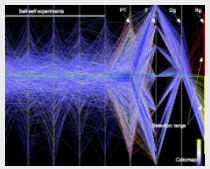


## Ästhetikkriterien I

Ästhetische Kriterien formulieren Wünsche an die Zeichnung eines Graphen, die die Zeichnung möglichst gut erfüllen soll;

- Oft sind nicht alle Kriterien, manchmal keines zu erreichen; es wird ein **guter Kompromiss** gesucht.

- **Kantenschnitte**. Minimierung der Anzahl der Schnittpunkte von Kanten
- **Fläche**. Graph wird auf möglichst kleiner Fläche gezeichnet. Dabei wird entweder das Gitterzeichnen oder ein Mindestabstand von Knoten vereinbart.
- **Gesamtkantenlänge**. Gesamtkantenlänge wird minimiert. Auch dies ist nur bei Gitterzeichnen oder Knotenmindestabstand sinnvoll.
- **Maximale Kantenlänge**. Länge der längsten Kante wird minimiert. Wieder muss man Gitterzeichnen oder Knotenmindestabstand vereinbaren.
- **Gleichförmige Kantenlänge**. Varianz der Kantenlängen wird minimiert.
- **Gesamtanzahl Knicke**. Zahl aller Knicke aller Kanten soll minimiert werden. Dies ist besonders bei orthogonalem Zeichnen sinnvoll.



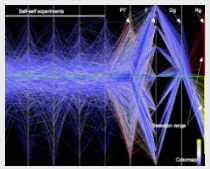


## Ästhetikkriterien II

- **Maximale Knickanzahl.** Maximum von Knicken einer Kante wird minimiert.
- **Gleichmäßige Knickanzahl**. Varianz der Knickanzahl wird minimiert.
- **Winkelauflösung**. Kleinster auftretender Winkel zwischen zwei Kanten wird maximiert.
- **Seitenverhältnis**. Seitenverhältnis des kleinsten umschriebenen Rechtecks soll nahe 1 sein.
- **Symmetrie**. Symmetrien des Graphen sollen möglichst gut abgebildet werden.

**Aber:**

- Ästhetikkriterien widersprechen sich oft.
- Einzelne Kriterien können nicht immer erfüllt werden.



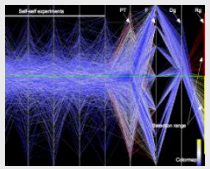


**Einschränkungen**

Festlegen von Teilgraphen oder Teilzeichnungen, um bestimmte Konventionen einzuhalten

Häufige Beispiele in der Visualisierung

- **Zentrum**. Gegebener Knoten wird im Zentrum der Zeichnung fixiert.
- **Außen**. Bestimmter Knoten wird der äußeren Facette benachbart.
- **Cluster**. Gegebene Menge von Knoten wird nah beieinander angeordnet.
- **Links-Rechts-Folge**. Vorgegebener Weg wird von links nach rechts gezeichnet (auch vertikal möglich).
- **Form**. Zeichne gegebenen Teilgraphen mit vorgegebener Form.





## Topologie-Form-Metrik-Ansatz

Dieser Ansatz zielt vor allem auf orthogonale Gitterzeichnungen und basiert auf drei Verfeinerungsebenen:

**Topologie**. Zwei orthogonale Zeichnungen haben **gleiche Topologie**, wenn sie durch

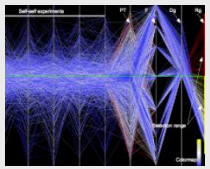
- **stetige Deformation ohne Vertauschen von Facetten** ineinander überführt werden können

**Form**. Zwei orthogonale Zeichnungen haben **gleiche Form**, wenn sie

- **gleiche Topologie** haben und
- eine Zeichnung aus der anderen nur durch **Längenänderungen der Liniensegmente** hervorgeht.

**Metrik**. Zwei orthogonale Zeichnungen haben **gleiche Metrik**, wenn sie

- **kongruent** sind, also durch
- **Verschieben und Drehen** ineinander überführt werden können.





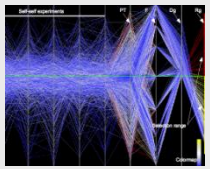
## Topologie-Form-Metrik-Ansatz/2

Umsetzung erfolgt durch drei entsprechende Schritte, bei denen nacheinander **Topologie**, Form und Metrik optimiert werden:

**Planarisierung** legt die Einbettung des Graphen, also seine **Topologie** fest:

- **Minimierung der Anzahl der Kantenschnitte**:
  - Oft wird zunächst ein **maximaler planarer Untergraph** gesucht und eingebettet.
  - Anschließend werden **verbleibende Kanten einzeln eingefügt**, wobei jeweils möglichst wenig Überschneidungen erzeugt werden.
  - An Überschneidungen werden **Dummy-Knoten eingefügt.**

[Jeron, Jard. 3D Layout of Reachability Graphs of Communicating Processes. Graph Drawing '94 Proceedings, 1995, 25-32],
[Jünger, Leipert, Mutzel. Pitfalls of using PQ-trees in automatic graph drawing. Graph Drawing '97 Proceedings, 1997, 193-204]
[Battista, Eades, Tamassia, Tollis. Graph Drawing. Prentice-Hall, 1999]





## Topologie-Form-Metrik-Ansatz/3

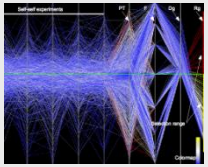
**Orthogonalisierung** legt die **Form** fest:

- **Orthogonale Repräsentation** des Graphen
  - Knoten **ohne Koordinaten**
  - Kanten lediglich **Winkellisten**, zur Festlegung ihrer Knicke
- **Minimierung der Knicke** ist in der Regel das Hauptziel

**Kompaktifizierung** legt endgültige **Koordinaten** der Kanten und der **Kantenknicke** fest:

- In der Regel wird die **Fläche des Graphen minimiert.**

[Jeron, Jard. 3D Layout of Reachability Graphs of Communicating Processes. Graph Drawing '94 Proceedings, 1995, 25-32],
[Jünger, Leipert, Mutzel. Pitfalls of using PQ-trees in automatic graph drawing. Graph drawing '97 Proceedings, 1997, 193-204]
[Battista, Eades, Tamassia, Tollis, Graph Drawing, Prentice-Hall, 1999]



# 5.2 Paradigmen

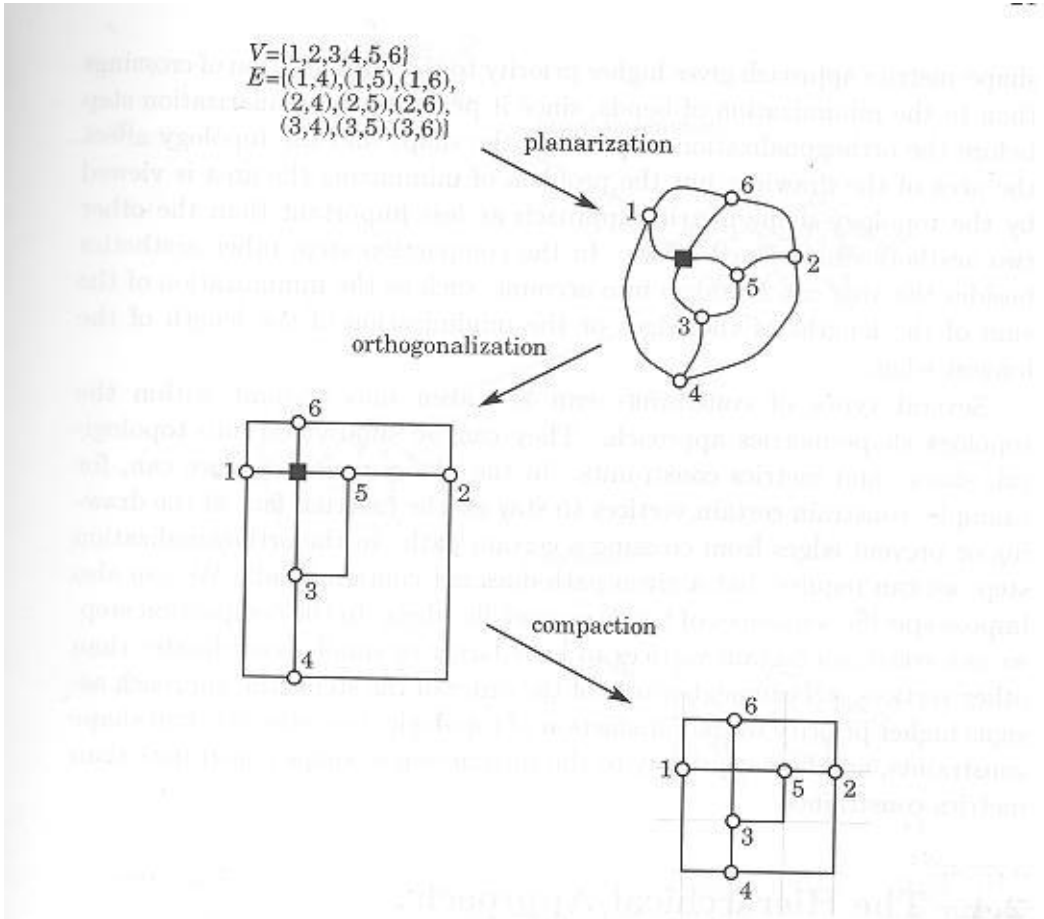
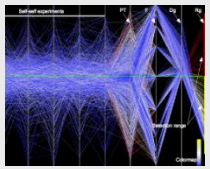



**Figure 2.6**: The topology-shape-metrics approach for orthogonal grid drawings. The dummy vertex introduced by the planarization step is represented by a square.

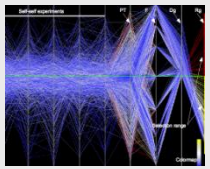




## Hierarchischer Ansatz

Ansatz eignet sich besonders für Layout von DAGs und verwendet ebenfalls drei Schritte

1. **Schichtungsschritt** verwandelt einen DAG in einen **geschichteten Digraphen**, bei dem jedem Knoten eine Schicht $L_i$ zugeordnet wird, so dass für alle Kanten (u,v) gilt: $u \in L_i \wedge v \in L_j$ und $i<j$
   - Nach **Einfügen von Dummy-Knoten** auf Kanten, die über mehr als eine Ebene laufen - so dass jede Kante von einer Schicht i zur Schicht i+1 läuft - ist die ein **echt geschichteter Digraph.**
2. Bei **Schnittreduzierung** erhält jede Schicht eine **Reihenfolge** (Ordnung)
   - Reihenfolge bestimmt **Topologie** des Layouts.
   - Dabei wird Anzahl der **Kantenschnitte minimiert.**



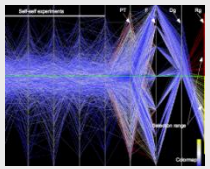


## Hierarchischer Ansatz/2

3. **Festlegen der x-Koordinaten** legt **x**-Koordinaten der Knoten in den Ebenen fest
   - Für endgültige Zeichnung ersetzt man Dummy-Knoten durch Knicke
   - **Knicke können minimiert** oder **Symmetrien betont** werden

Für **Digraphen** wird minimale Anzahl von Kanten umgedreht, um einen DAG zu erzeugen

Diese werden aber in ihrer ursprünglichen Richtung gezeichnet.



# 5.2 Paradigmen

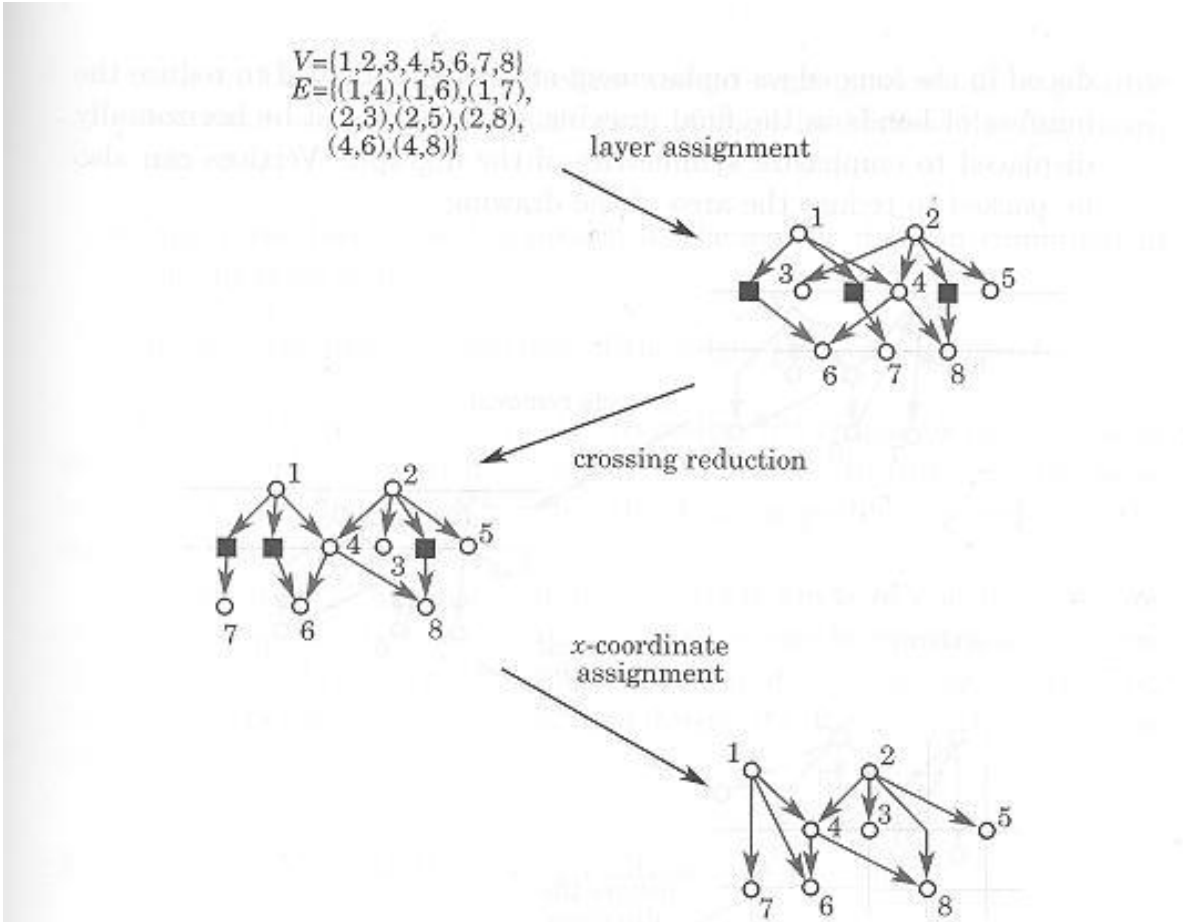
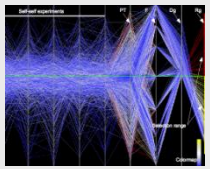



**Figure 2.7**: The hierarchical approach. The dummy vertices introduced in the layer assignment step are represented by squares.



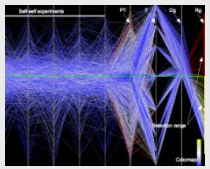
# 5.2 Paradigmen

- Problem der **Schnittminimierung** zwischen zwei Ebenen ist bereits **NP-hart**: Heuristiken werden gesucht
  [K. Sugiyama, S. Tagawa, and M. Toda, „Methodes for Visual Understanding of Hierarchical Systems Structures“, IEEE Trans. Systems, Man, and Cybernetics, vol. 11, no. 2, pp. 109-125, 1989]

- Heuristik legt eine **Ordnung der ersten und letzten Ebene fest:** jeder Knoten soll im Schwerpunkt seiner Nachbarn (im Graphen) liegen: lineares Gleichungssystem

  [W. Tutte, „How to Draw a Graph“, Proc. London Math. Soc., vol. 3, no. 13, pp. 743-768, 1963]

- Vergleich von Heuristiken:
  [M. Laguna and R. Marti, „Heuristics and Meta-Heuristics for 2-Layer Straight Line Crossing Minimization“, URL: http://www-bus.colorado.edu/Faculty/Laguna/, 1999].

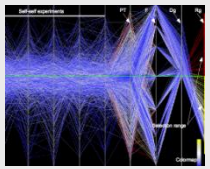




## Sichtbarkeitsansatz

Ansatz dient dem Auslegen beliebiger Graphen mit Polylinien

- **Planarisierung** gleicht dem Topologie-Form-Metrik-Ansatz; Ziel ist **Schnittvermeidung**
- **Sichtbarkeitsschritt** schafft eine Sichtbarkeitsrepräsentation des Graphen
  - Jedem **Knoten** wird **horizontales** und jeder **Kante** {u,v} **vertikales** Segment zugeordnet
  - Kantensegment hat Anfang und Ende innerhalb Knotensegmente
  - Kantensegmente schneiden einander nicht
- **Ersetzungsschritt** erzeugt endgültige Zeichnung des Graphen
  - Ersetzen jedes **horizontalen** Segments durch **Punkt**
  - Ersetzen jedes **vertikalen** Segments durch **Polylinie**
  - Mögliche Strategien / Ästhetikkriterien: Knickminimierung, Symmetriebetonung, gleichmäßige Verteilung der Knoten



# 5.2 Paradigmen

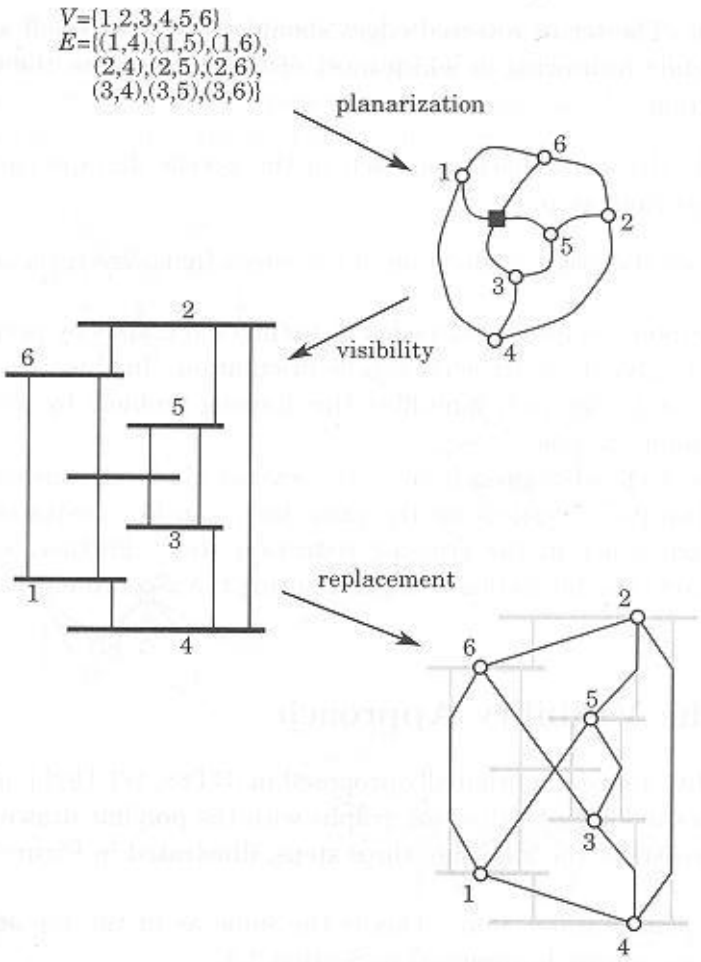



Figure 2.9: The visibility approach.

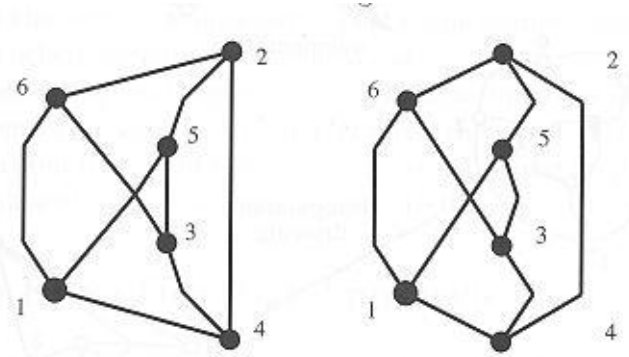
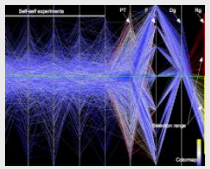


Figure 2.10: Different replacement strategies in the visibility approach.

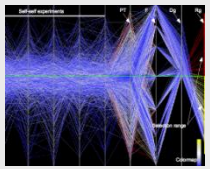




## Verfeinerungsansatz

Betrachtet Polylinienzeichnungen

- **Planarisierung** gleicht Topologie-Form-Metrik-Ansatz;
  Hier geht es erneut vor allem um Schnittvermeidung
- **Verfeinerung** fügt geeignete Anzahl von Kanten (evtl. auch Knoten) in Einbettung ein, um einen **maximalen planaren Graphen** zu erzeugen:
  - Alle Facetten haben dann drei Kanten - **Triangulierung**
  - Qualität des Zeichnens maximaler Graphen hängt in der Regel vom **Knotengrad** ab
    - Minimierung des **maximalen Knotengrad**



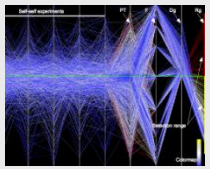


## Verfeinerungsansatz/2

Betrachtet Polylinienzeichnungen

- **Triangulierungszeichnung** erzeugt Zeichnung durch
  - **Auslegen aller Dreiecke** und
  - anschließendes **Entfernen der Dummy-Kanten** und ggf. Dummy-Knoten

- Alle Algorithmen nutzen spezielle **Triangulierungseigenschaften** durch
  - überschneidende aufspannende Bäume oder
  - kanonische Knotenaufzählungen
    (Beim Entfernen der Dummy-Knoten können Knicke und Schnitte entstehen.)



# 5.2 Paradigmen

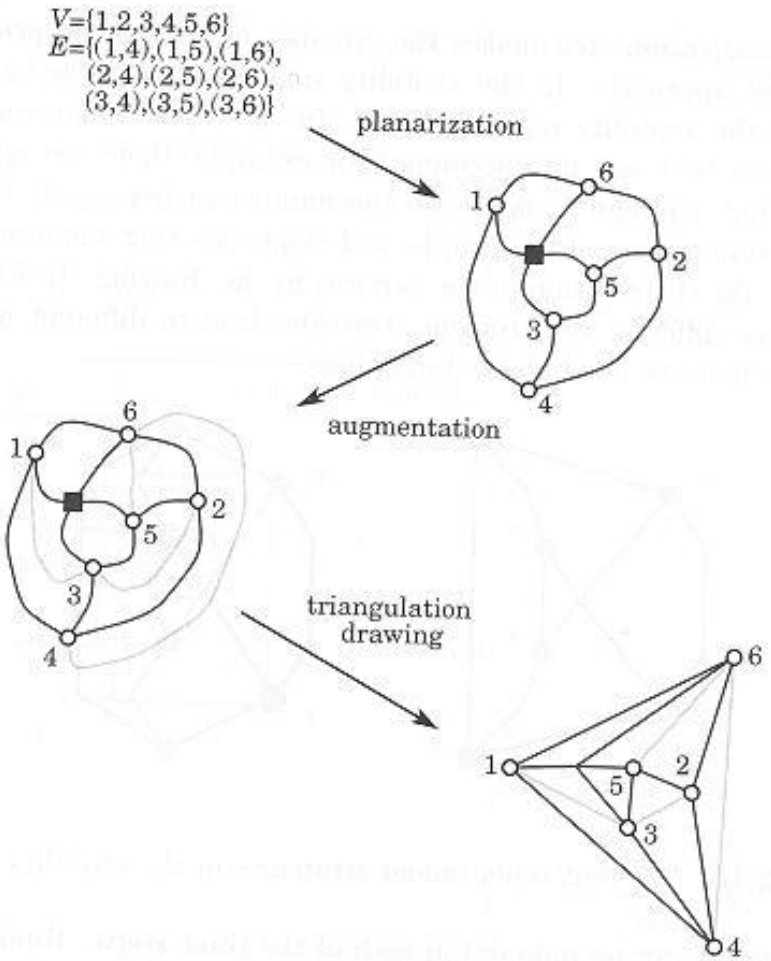
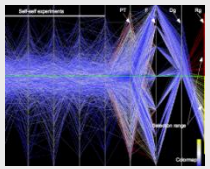



**Figure 2.11**: The augmentation approach.

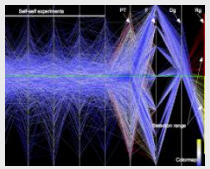




## Kraftbasierter Ansatz

- Kraftbasierte Ansätze sind intuitive Methoden für geradlinige Zeichnungen ungerichteter Graphen.
- Sehr populär wegen ihrer **leichten Verständlichkeit** und **einfachen Implementierung**
- Simuliert System von Kräften auf Eingabegraphen und gibt Zeichnung minimaler Energie aus. Dazu braucht man:
  - **Kräftemodell**: Oft wird jedem Paar von Knoten eine **Feder „natürlicher Länge“** zugeordnet, etwa Anzahl der Kanten des kürzesten Weges
    - Federn gehorchen **Federgesetz** und
    - Kraft proportional zur **Differenz** von natürlicher Länge und Länge in aktuellen Zeichnung
  - **Energieminimierung**: einfache **iterative Ansätze** zur Lösung gewöhnlicher Differentialgleichungen

[A. Frick, A. Ludwig, and H. Mehldau, „A Fast Adaptive Layout Algorithm for Undirected Graphs“, Proc. Symp. Graph Drawing (GD '94), pp. 389-403, 1994]

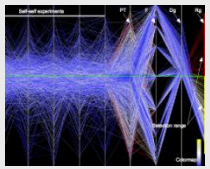




## Kraftbasierter Ansatz/2

- Kraftbasierte Ansätze sind intuitive Methoden für geradlinige Zeichnungen ungerichteter Graphen
- Sehr populär wegen ihrer **leichten Verständlichkeit** und **einfachen Implementierung**
- Simuliert System von Kräften auf Eingabegraphen und gibt Zeichnung minimaler Energie aus. Dazu braucht man:
  - **Kräftemodell**
  - **Energieminimierung**
  - Ergibt oft **qualitativ gute, hoch symmetrische** Zeichnungen
    - mit gleichmäßiger Knotenverteilung
    - wobei Einschränkungen können beachtet werden

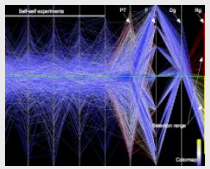
[A. Frick, A. Ludwig, and H. Mehldau, „A Fast Adaptive Layout Algorithm for Undirected Graphs“, Proc. Symp. Graph Drawing (GD '94), pp. 389-403, 1994]





## Divide-and-Conquer Ansatz

- Beliebtes Informatikprinzip wird auch bei Zeichnungen von Graphen gerne benutzt
  - **Zerlegt** Graph in **Teilgraphen**, zeichnet diese und erhält letztlich eine Zeichnung des gesamten Graphen
  - Besonders eignen sich hier Bäume





## Übersicht

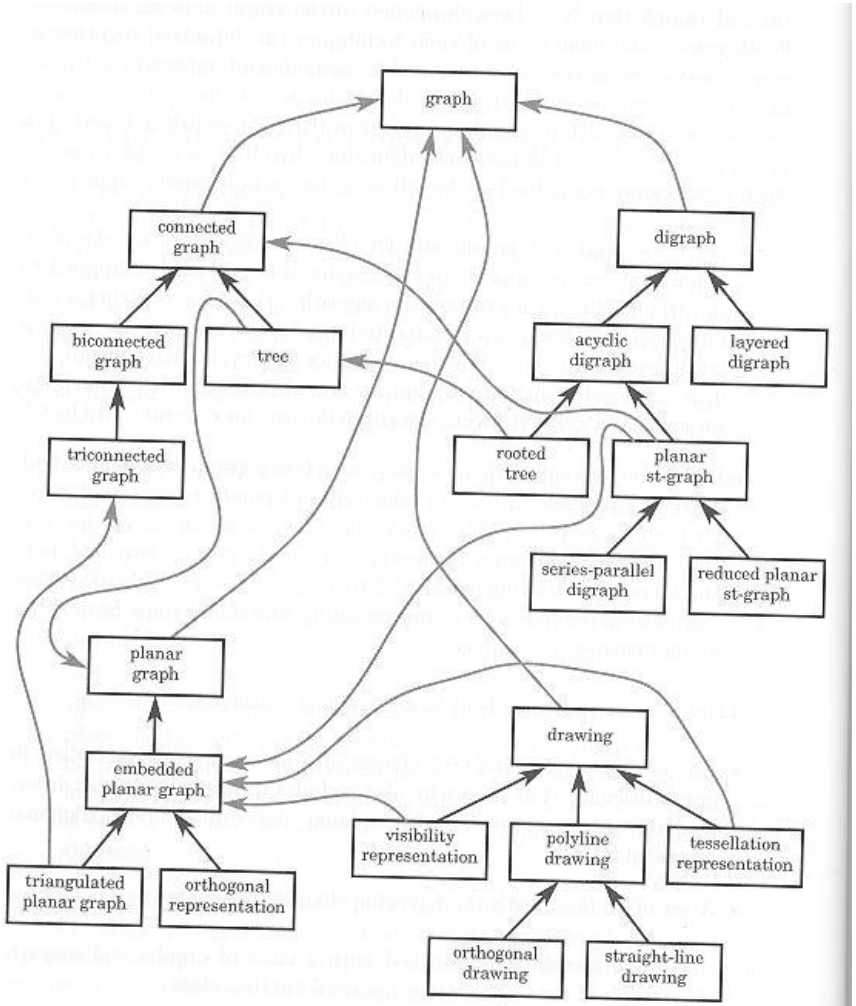
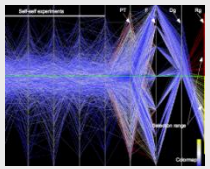



**Figure 2.12**: Inheritance hierarchy of classes of graphs, drawings, and related representations.

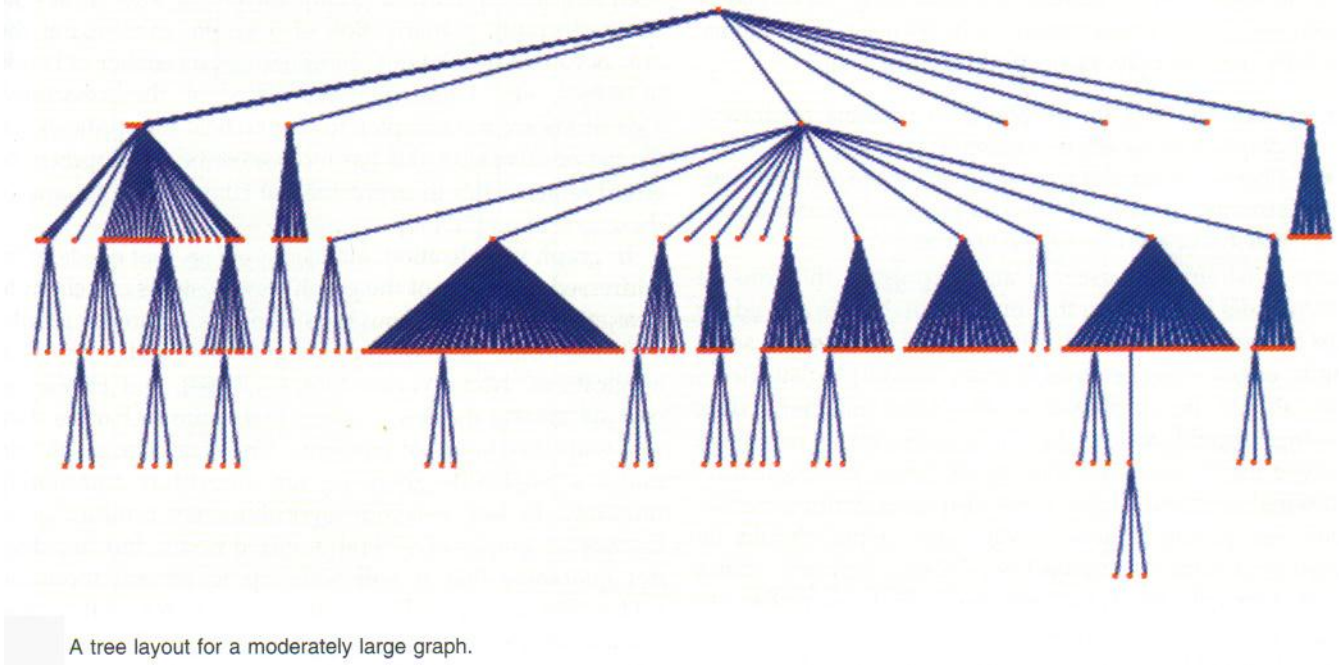




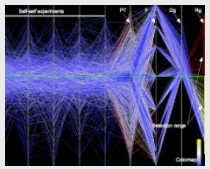
## Reingold / Tilford Algorithmus

- Erzeugt **klassische Baumstruktur** für Bäume mit ausgezeichneter Wurzel mit einem typischen Divide-and-Conquer Ansatz
- Alle Knoten einer Ebene werden auf **gleiche Höhe** gelegt
- Horizontale Raum wird zwischen Teilbäumen entsprechend der **Anzahl ihrer Blätter** aufgeteilt

[E. M. Reingold and J. S. Tilford, „Tidier Drawing of Trees“, IEEE Trans. Software Eng., 7(2), pp. 223-228, 1981]



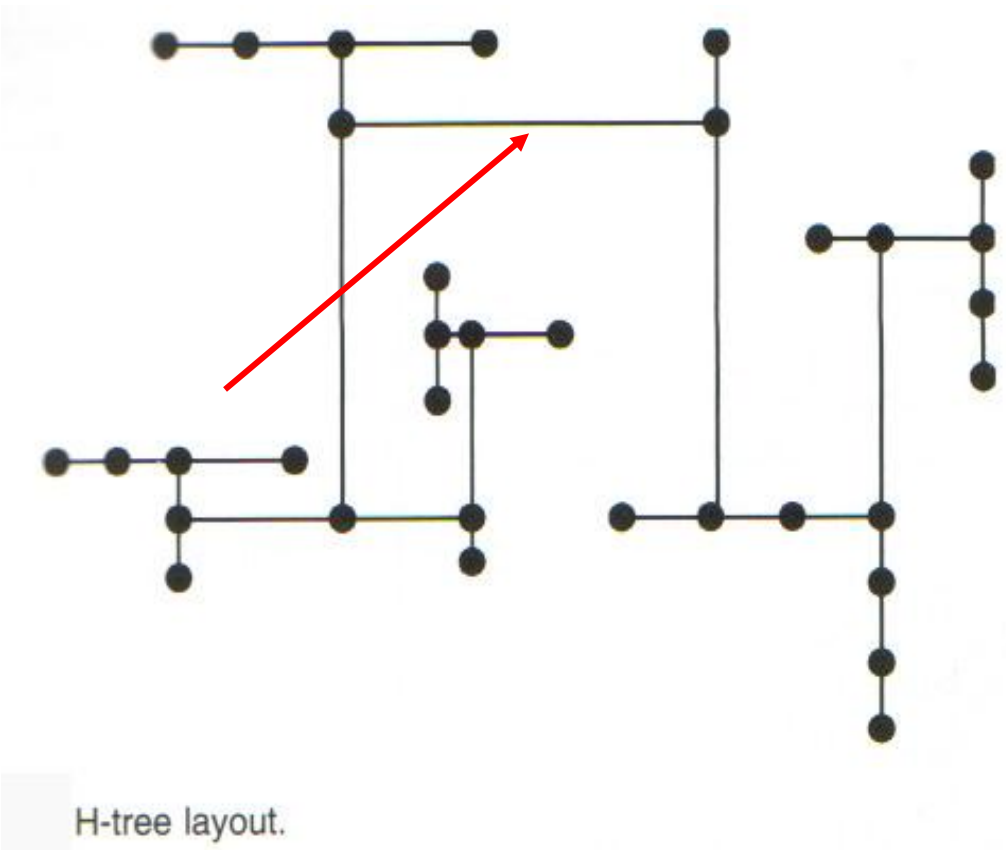
A tree layout for a moderately large graph.



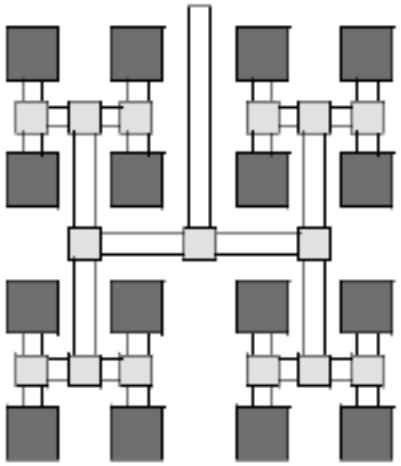
# 5.3 Divide and Conquer Methoden

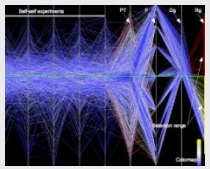
- Bäume können auch **H-förmig ausgelegt** werden
- Wird für Chip-Layout verwendet

[P. Eades, „Drawing Free Trees“, Bulletin of the Inst. For the Combinatorics and Its Applications, pp. 10-36, 1992].



H-tree layout.



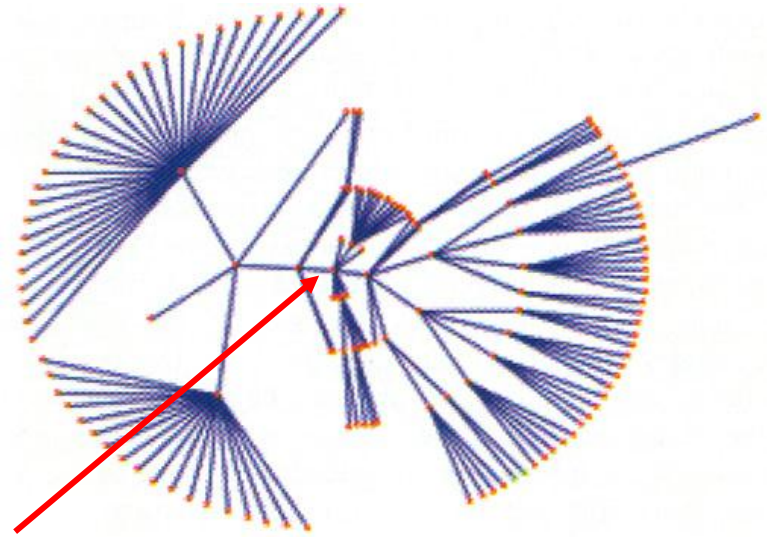


# 5.3 Divide and Conquer Methoden

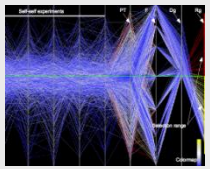
Weitere Divide-and-Conquer Variante:

- Radiales Auslegen des Baums mit **Wurzel im Zentrum**
- Alle Knoten einer Ebene liegen auf **konzentrischen Kreisen**
- **Algorithmus vermeidet Überschneidungen** durch Festlegen der Sektoren für die Teilbäume
  - Man kann letzte Bedingung abschwächen, um im Mittel gute Ergebnisse zu erhalten

[I.Herman, G. Melancon, M. M. De Ruiter, and M. Delest, „Latour-A Tree Visualization System“, Proc. Symp. Graph Drawing GD'99, pp. 392-399, 19
A more detailed version in: Reports of the Centre for Math. And Computer Science, Report number INS-R9904, available at:
http://www.cwi.nl/InfoVisu/papers/LatourOverview.pdf, 1999]



Radial view.

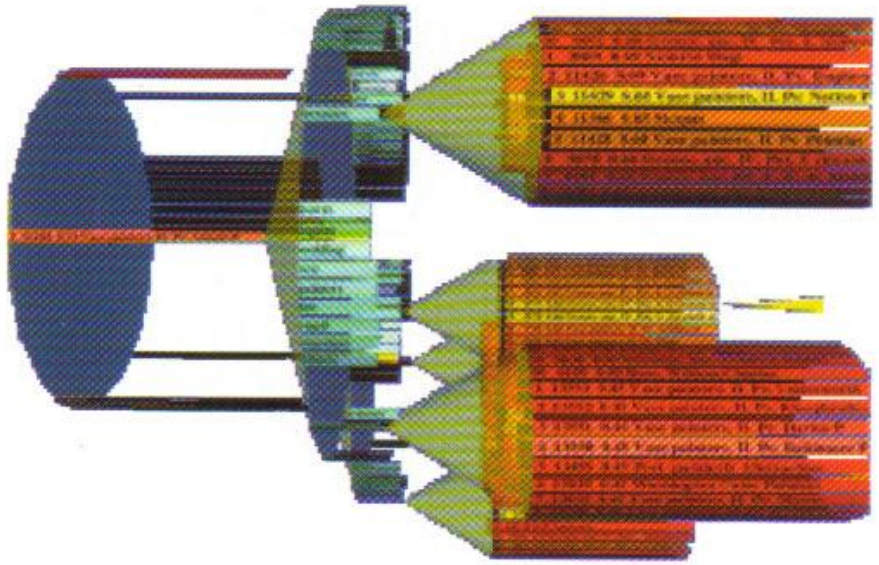


# 5.3 Divide and Conquer Methoden

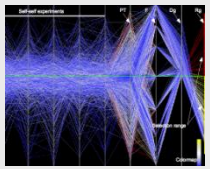
## Kegelbaum (cone tree):

- Geschwister werden in Kreis ausgelegt
- Ergibt mit Elterknoten einen Kegel
- Divide-and-Conquer Ansatz für Bäume mit Wurzel
  - Allerdings keine Zeichnung wegen 3D-Layouts

[J. Carriere and R. Kazman, „Research Report: Interacting with Huge Hierarchies: Beyond Cone Trees“, Proc. IEEE Information Visualization '95, pp. 74-81, 1995]



A cone tree. (Courtesy of M. Hemmje, GMD, Germany [59].)

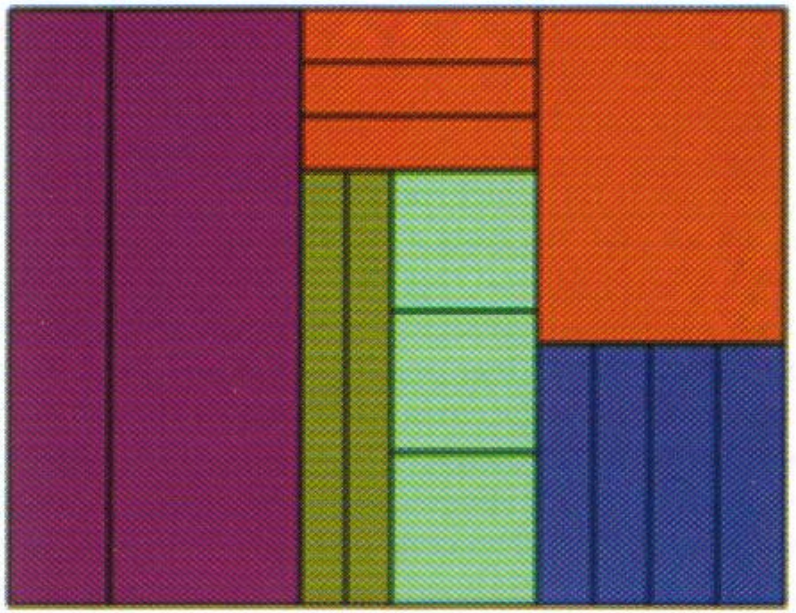




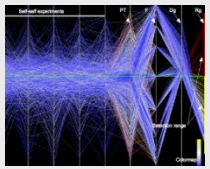
## Baumkarte (tree map):

- Jedem **Teilbaum wird ein Rechteck** zugewiesen
- Rechteck wird weiter unterteilt
- Gestattet Visualisierung weiterer **Information durch die Größe der Rechtecke** (und Farbe, was aber bei praktisch allen Zeichnungen möglich ist)

[B. Johnson and B. Schneiderman, „Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures“, Proc. IEEE Visualization '91, pp. 275-282, 1991]



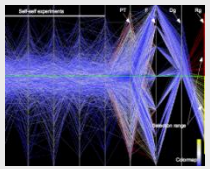
Tree-map: rectangles with color belong to the same level of the (tree) hierarchy. (Adapted from Johnson and Schneiderman [72]).



# 5.3 Divide and Conquer Methoden

- Alle besprochenen Baumauslegealgorithmen sind **vorhersagbar**
  - Bei **gleichem Input** liefern sie gleichen Output
  - **Isomorphe Teilbäume** werden **gleich behandelt**
  - **sehr sinnvolle Eigenschaft für** Visualisierung

- **kraftbasierte Ansätze** weisen Eigenschaft **nicht** mehr auf
  - Meistens gibt es **mehrere lokale Minima**



# 5.4 Kraftbasierte Methoden

- **Kraftbasierte Methoden** orientieren sich an einem physikalischen Modell mit Kräften bzw. potentieller Energie
- Energie wird minimiert
- Relativ einfach zu implementieren
- Sehr beliebt, da intuitiv zu verstehen und zugleich oft gute Ergebnisse liefernd

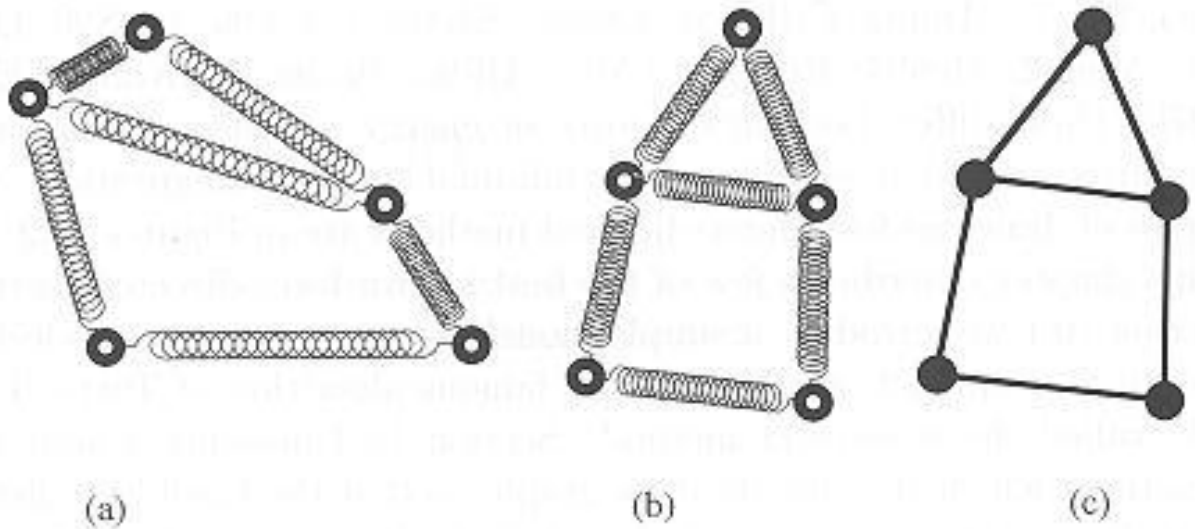
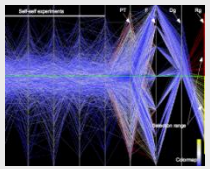



Figure 10.1: A spring algorithm.





## Federn und elektrische Kräfte

Einfachster Ansatz nutzt **elektrische Kräfte und Federn**

- Jeder **Knoten v als positiv geladenes Teilchen** und jede **Kante e={u,v} als Feder** mit vorgegebener Ruhelänge $l_{uv}$
- Kraft auf einen Knoten v ist

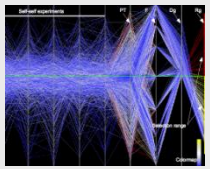
$$F(v) = \sum_{\{u,v\}} f_{uv} + \sum_{(u,v)\in V\times V} g_{uv}$$

wobei $f_{uv}$ die Kraft auf v durch **die Feder zwischen u und v ist** und $g_{uv}$ **elektrische Abstoßung von v durch u** modelliert

Genauer gilt

$$F(v) = \sum_{\{u,v\}} k_{uv}^{(1)}\left(d(p_u,p_v) - l_{uv}\right)\frac{p_u - p_v}{d(p_u,p_v)} + \sum_{(u,v)\in V\times V} \frac{k_{uv}^{(2)}}{\left(d(p_u,p_v)\right)^2}\frac{p_u - p_v}{d(p_u,p_v)}$$

mit Zeichnungspositionen $\mathbf{p_u}$, $\mathbf{p_v}$, euklidischen Abstand $\mathbf{d(p_u,p_v)}$, den Federkonstanten $\mathbf{k_{uv}^{(1)}}$, ladungsbasierten Abstoßungsstärke $\mathbf{k_{uv}^{(2)}}$



# 5.4 Kraftbasierte Methoden

- Lösung mit **numerischen Verfahren** mit häufig genutztem intuitiven Ansatz
  - Plaziere Knoten **zufällig in Ebene**
  - **Berechne Kraft** F(v) für alle Knoten v
  - **Bewege jeden Knoten** v ein kleines Stück in Richtung F(v)
  - Wenn **Kräfte nicht annähernd Null** sind und das Maximum an **Iterationen nicht erreicht** ist, gehe zu (2)

- Alternative, schnellere Ansätze liefert beispielsweise **Numerik gewöhnlicher Differentialgleichungen**

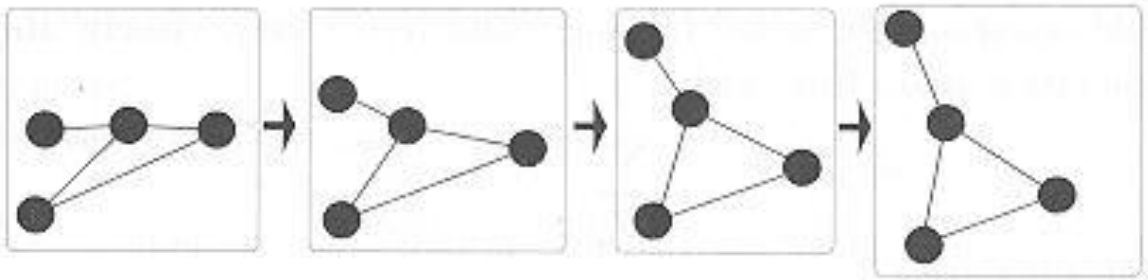
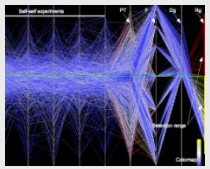


Figure 10.8: Frames in an animation of a spring algorithm.



# 5.4 Kraftbasierte Methoden

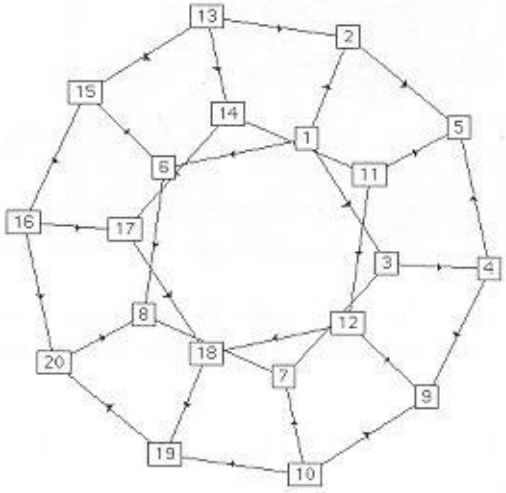



**Figure 10.3**: Dodecahedron drawn using a force-directed algorit
tesy of U. Erlingsson and M. Krishnamoorthy.)

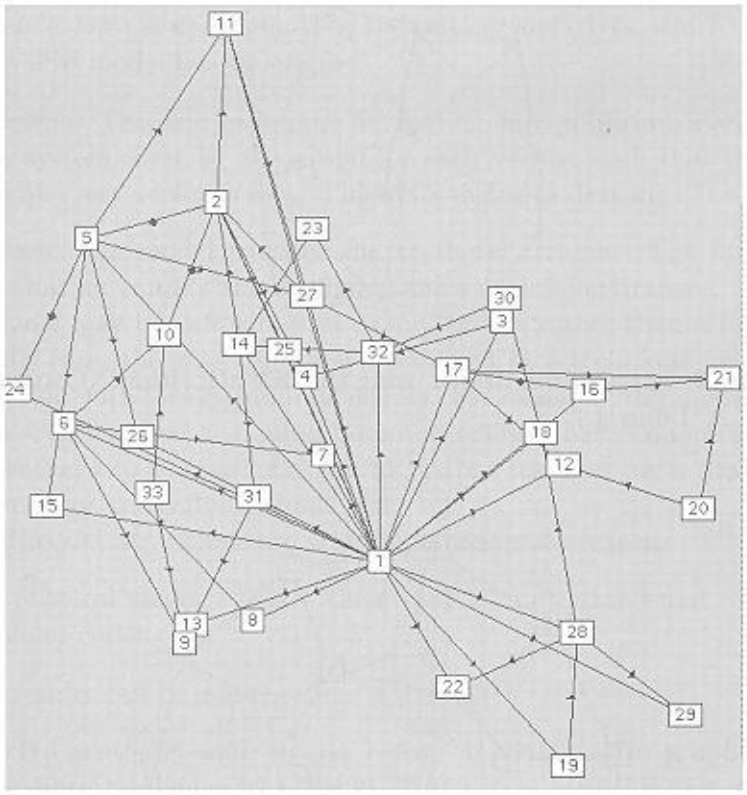
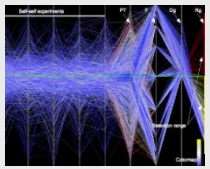



**Figure 10.4**: Graph of internet traffic drawn using a simple spring algorithm. (Courtesy of J. Fenwick, D. Thompson and R. Stacey.)



# 5.4 Kraftbasierte Methoden

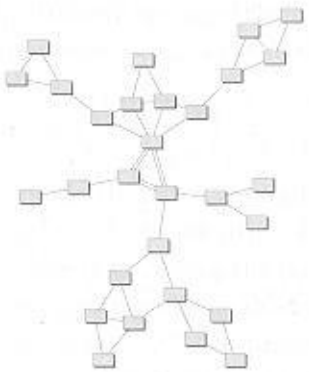


**Figure 10.5**: Graph drawn using a simple spring algorithm. (Courtesy of Tom Sawyer Software.)

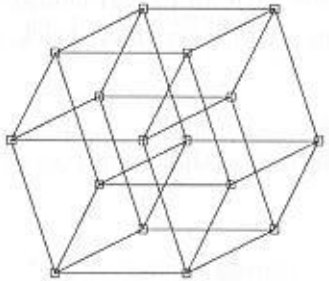


**Figure 10.6**: A hypercube drawn with the method of Tunkelang. (Courtesy of D. Tunkelang.)

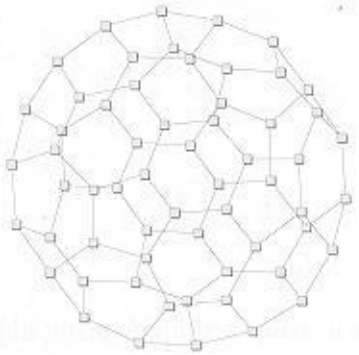
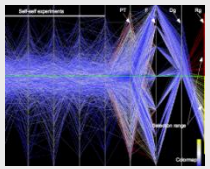


**Figure 10.7**: Graph drawn with an experimental force-directed algorithm from Tom Sawyer Software. (Courtesy of A. Frick.)





## Baryzentrische Methode

Nutzt Federn der Länge 0, verzichtet auf elektrische Kräfte, **fixiert aber eine ausgewählte Menge von Knoten** als konvexen Rand der Zeichnung.

Es ergibt sich

$$F(v) = \sum_{\{u,v\} \in E} (p_u - p_v)$$

Mit **$N_0(v)$ als fixierte Nachbarn von v** und mit **$N_1(v)$ als freie Nachbarn** ergeben sich linearen Gleichungen

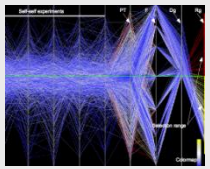
$$deg(v) p_v - \sum_{u \in N_1(v)} p_u = \sum_{w \in N_0(v)} p_w \quad \text{für nicht fixierte } v \in V$$

die nichts anderes sagen, als dass jeder Knoten als **Schwerpunkt** (Barycenter) seiner Nachbarn gezeichnet wird

$$p_v = \frac{\sum_{u \in N_1(v)} p_u + \sum_{w \in N_0(v)} p_w}{deg(v)}$$

für nicht fixierte $v \in V$

[Tutte. Convex Representation of Graphs. Proc. of London Mathematical Society 10(3):304-320,1960],
[Tutte. How to Draw a Graph, Proc. of London Mathematical Society 13(3):743-768, 1963]



# 5.4 Kraftbasierte Methoden

**Algorithm 10.1** *Barycenter-Draw*

*Input:* graph $G = (V, E)$; a partition $V = V_0 \cup V_1$ of $V$ into a set $V_0$ of at least three *fixed* vertices and a set $V_1$ of *free* vertices; a strictly convex polygon $P$ with $|V_0|$ vertices

*Output:* a position $p_v$ for each vertex of $V$, such that the fixed vertices form a convex polygon $P$

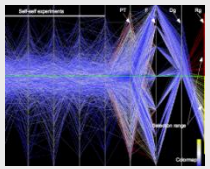
1. Place each fixed vertex $u \in V_0$ at a vertex of $P$, and each free vertex at the origin.
2. **repeat**

   **foreach** free vertex $v$ **do**

   $$x_v = \frac{1}{deg(v)} \sum_{(u,v)\in E} x_u$$
   $$y_v = \frac{1}{deg(v)} \sum_{(u,v)\in E} y_u$$

   **until** $x_v$ and $y_v$ converge for all free vertices $v$.

□



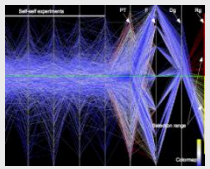


## Theorem

Sei

- G ein **dreifach zusammenhängender Graph**,
- sei f eine **Facette einer planaren Einbettung** von G und
- sei P eine **streng konvexe Zeichnung** von f,

dann liefert vorhergehende Algorithmus eine **konvexe, planare Zeichnung von G mit fixierten Ecken von f**



# 5.4 Kraftbasierte Methoden

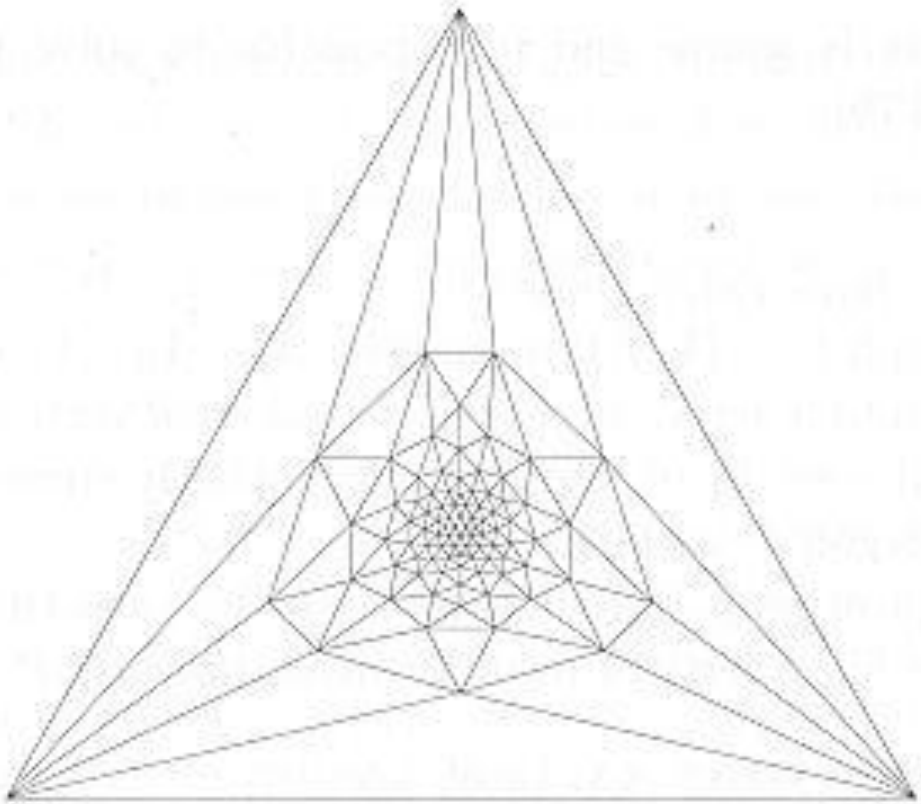
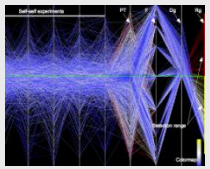


**Figure 10.9**: A large planar graph drawn with the barycenter method. (Courtesy of P. Garvan.)





## Auf graphentheoretischen Distanzen beruhende Kräfte

- Sei G=(V,E) ein zusammenhängender Graph. Für zwei Knoten **u,v** heißt die Länge **δ(u,v)** des kürzesten Weges **graphentheoretische Distanz**
- Ziel dieser Ansätze ist Übereinstimmen von **euklidischer Distanz** in der Zeichnung und **graphentheoretischer Distanz** im Graphen
- Kamada und Kawai wählen daher $k_{uv} := k/\delta(u,v)^2$ mit Konstante k

und setzen die **potentielle Energie** der Feder zur Kante {u,v} auf

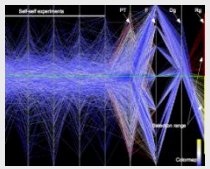
$$\eta_{\{u,v\}} = \frac{k}{2}\left\{\frac{d(p_u,p_v)}{\delta(u,v)} - 1\right\}^2.$$

sowie der gesamten Zeichnung auf

$$\eta = \frac{k}{2}\sum_{u \neq v \in V}\left\{\frac{d(p_u,p_v)}{\delta(u,v)} - 1\right\}^2$$

- Dieser Wert wird dann **iterativ minimiert**
  - Nach allen Knotenpositionen ableiten
  - Knoten in Richtung mit größter Ableitung bewegen

[Kamada, Kawai. An Algorithm for Drawing Undirected Graphs. Information Processing Letters 31(1):7-15, 1989]





## Magnetische Felder

- Man kann Federn auch **magnetischen Kräften** aussetzen, um sie möglichst gleich auszurichten.
- Dabei werden einige oder alle **Federn im Modell magnetisiert** .
- Es wirkt ein **globales magnetisches Feld** in senkrechter, waagrechter oder radialer Richtung (mit Zentrum).

[Sugiyama, Misue. Graph Drawing by Magnetic Spring Mode Journal of Visual Language Computation 6(3), 1995].

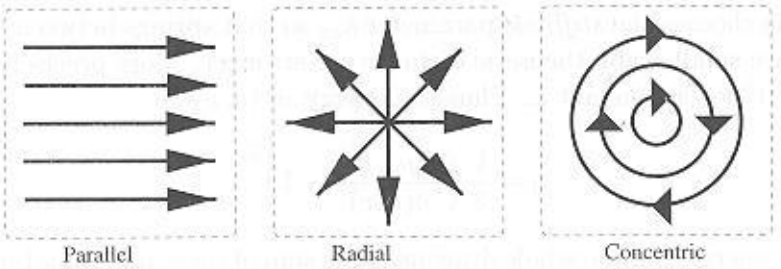



**Figure 10.10**: Types of magnetic field.

- Federn sind, wie folgt, modelliert:
  - **Unidirektional** mit Tendenz zur Ausrichtung in Feldrichtung, oder
  - **bidirektional** mit Ausrichtung nur parallel zum Feld

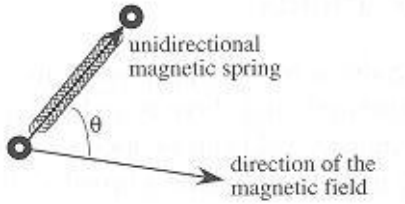
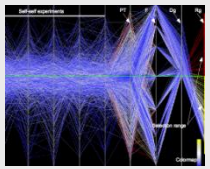



**Figure 10.11**: Magnetic spring.



# 5.4 Kraftbasierte Methoden

- **Unidirektionale Federn** und **magnetische Kräfte** erlauben Auslegen von Digraphen.

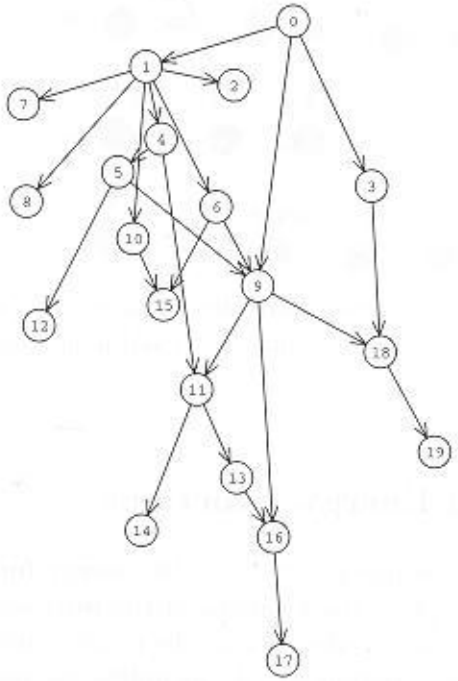


**Figure 10.12**: Magnetic spring drawing using a vertical magnetic field and unidirectional magnetic springs.

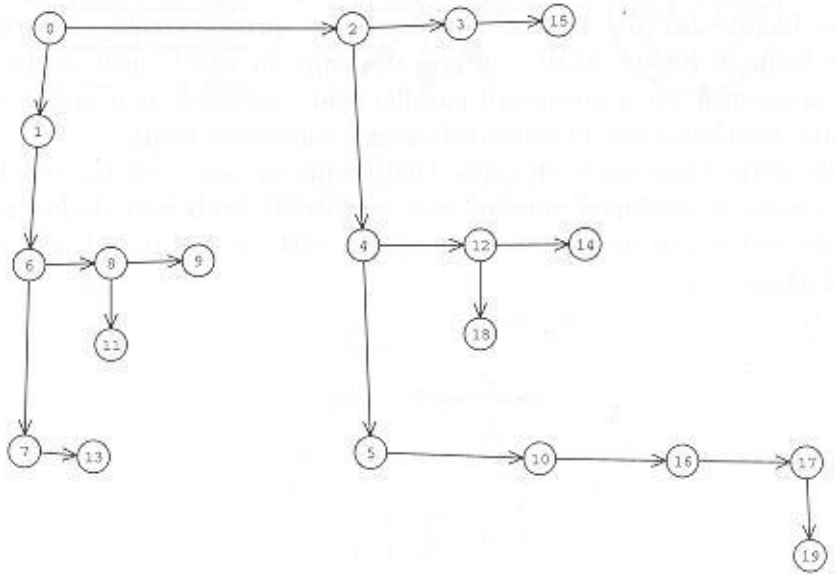
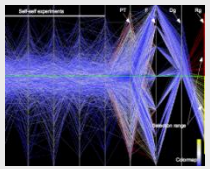


**Figure 10.13**: Magnetic spring drawing using a combination of horizontal and vertical magnetic fields and unidirectional magnetic springs.





## Allgemeine Energiefunktionen

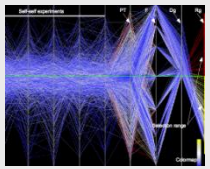
- Neben kontinuierlichen Energieansätzen können auch **diskrete Ästhetikmaß**e als Energien genutzt werden
  - Anzahl der Kantenschnitte
  - Anzahl der vertikalen und horizontalen Kanten
  - Anzahl der Knicke

Dabei verwendet man als Energie der Zeichnung

$$\eta=\lambda_1\eta_1+\ldots+\lambda_k\eta_k$$

wobei $\eta_i$, **i=1,...,k** einzelnen Kriterien messen

- Können auch folgende Energien enthalten: potentiellen Federenergien, potentiellen elektrischen und magnetischen Energien
- $\lambda_i$ sind **Gewichtungen** der Einzelenergien





## Beispiel allgemeiner Energien

Davidson / Harel verwenden vier Teile

$$\eta=\lambda_1\eta_1+\lambda_2\eta_2+\lambda_3\eta_3+\lambda_4\eta_4$$

Wobei $\eta_1 = \sum_{u,v \in V} d(p_u, p_v)^{-2}$ eine Art **elektrischer Abstoßung** simuliert,

- mit Randabständen $\mathbf{r_u}$, $\mathbf{l_u}$, $\mathbf{t_u}$, $\mathbf{b_u}$ das **Annähern an den Rand** bestraft,

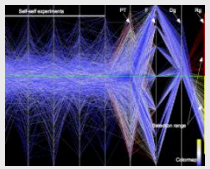
$$\eta_2 = \sum_{u \in V} \left( r_u^{-2} + l_u^{-2} + t_u^{-2} + b_u^{-2} \right)$$

- **zu lange Kanten** bestraft

$$\eta_3 = \sum_{\{u,v\} \in E} d(p_u, p_v)^2$$

- und $\eta_4$ **die Anzahl der Kantenschnitte** gering hält

[Davidson, Harel. Drawing Graphics Nicely Using Simulated Annealing. ACM Transactions on Graphics 15(4):301-331, 1996]





## Nachteile dieser Ansätze

- Finden **nicht immer** die beste Lösung
- Sind **stark parameterabhängig**

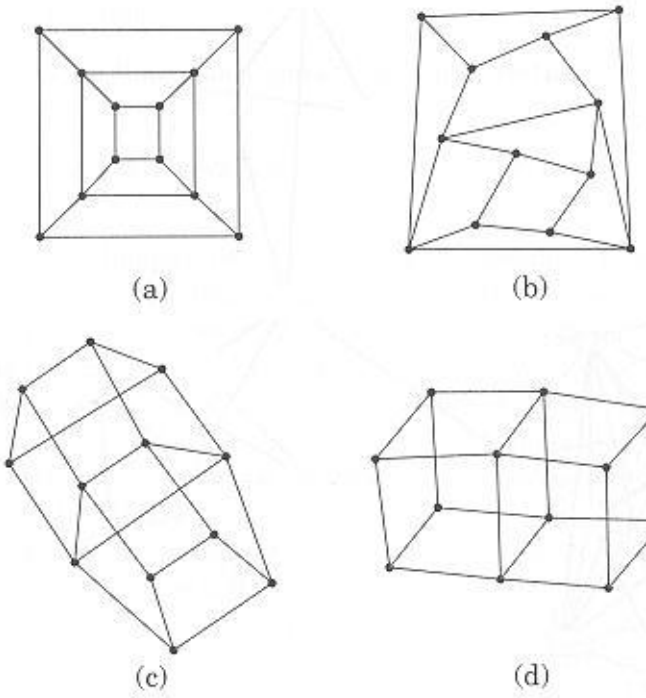
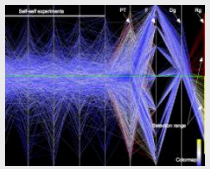



**Figure 10.16**: Drawings of a 12-vertex graph: (a) is an excellent drawing, not obtained by simluated annealing; (b), (c), and (d) were obtained using a general energy function, with various coefficients. The minimum energy state was found by a simulated annealing method. (Courtesy of David Harel.)



# 5.4 Kraftbasierte Methoden

- Aber viele gelungene Visualisierungsanwendungen

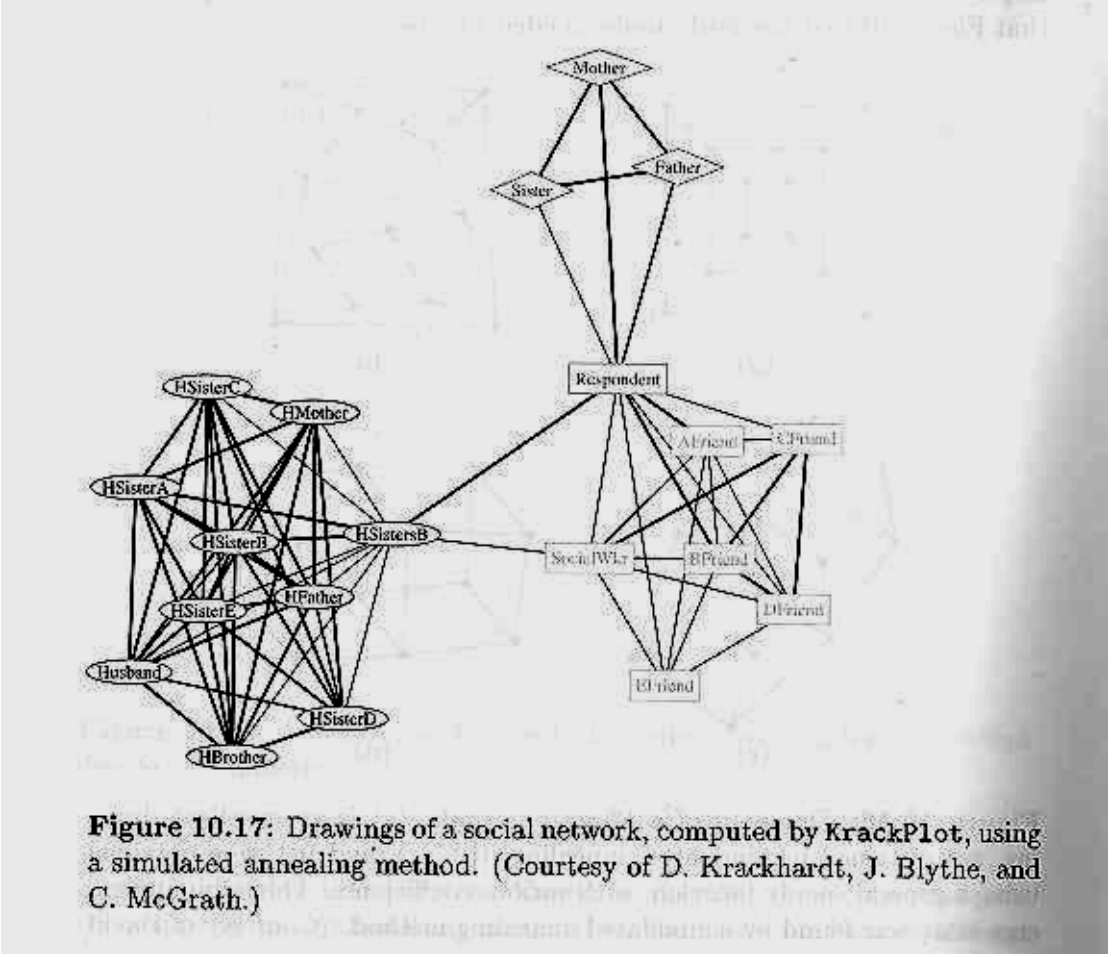
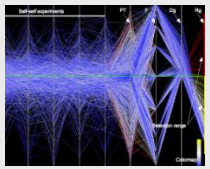


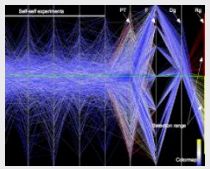

**Figure 10.17:** Drawings of a social network, computed by `KrackPlot`, using a simulated annealing method. (Courtesy of D. Krackhardt, J. Blythe, and C. McGrath.)





## Einschränkungen / Constraints

- Kraftbasierte Methoden lassen sich gut mit Einschränkungen wie
  - **festgelegte Positionen** einiger Knoten,
  - **fixierte** Teilgraphen
  - als Energie beschreibbare Beschränkungen
- verbinden



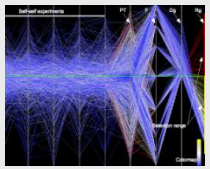


## Bemerkungen

- Kraftbasierte Methoden benötigen oft **viel Rechenaufwand**
- Daher **effiziente Minimierungsmethoden** nötig
  - Auf Zufallsbasis (Simulated Annealing, Behandlung steifer Differentialgleichungen, Kombinatorische Vorbehandelung)
  - Geeignete **Heuristiken**

- Weitere Details
  - Buch von Battista et al., der dort zitierten Literatur

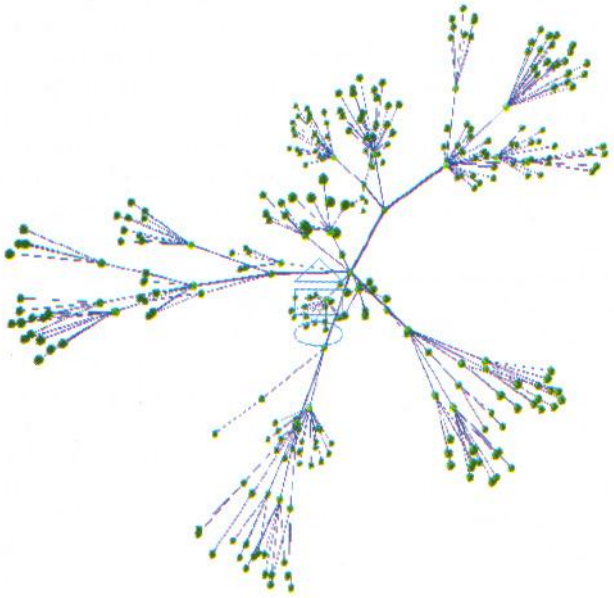
    [Battista, Eades, Tamassia, Tollis. Graph Drawing. Prentice Hall, Upper Saddle River, NJ, USA, 1999]
  - Proceedings der „Graph Drawing Conference Series“



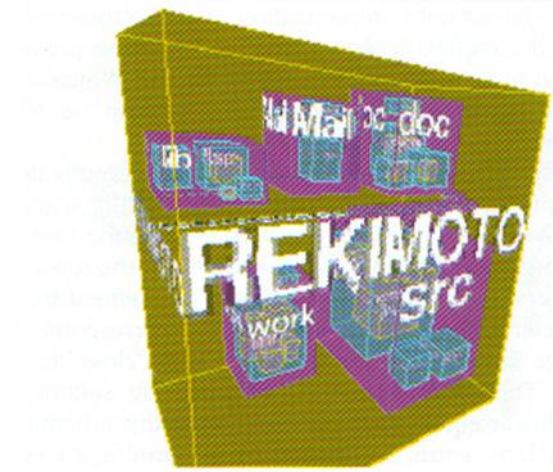


## 3D Layout

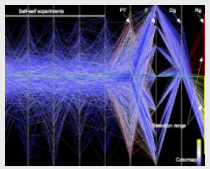
- Nutzung von 3D-Ansätze zum Layout: Zwei Beispiele (plus Kegelbaum)
- Allerdings sind Eingabegeräte (noch) nicht effizient zur 3D-Navigation nutzbar



3D version of a radial algorithm. (Courtesy of S. Benford, University of Nottingham, U.K.)



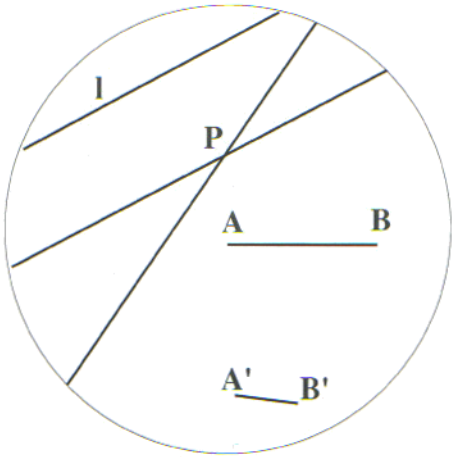

Information Cube. (Courtesy of J. Rekimoto, Sony Computer Science Laboratory, Inc., Japan [104].)



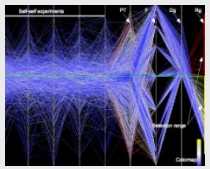


## Modelle der hyperbolischen Geometrie

- Sehr erfolgreich
- In dieser Geometrie gelten **Gesetze von Euklid**.
  - Aber: Zu einer Geraden gibt es **mehr als eine parallele Gerade** durch einen festen Punkt außerhalb der Geraden.
- Beim **Kleinmodell** (nach Felix Klein) wird Kreisscheibe in euklidischer Ebene benutzt.
  - Punkte liegen alle innerhalb
  - Geraden sind Sekanten (enden auf dem **Rand**)
  - Für hyperbolische Länge gibt es eine Formel, die Linienstücke gleicher hyperbolischer Länge im **euklidischen Sinne zum Rand** der Scheibe **immer kürzer** werden lässt

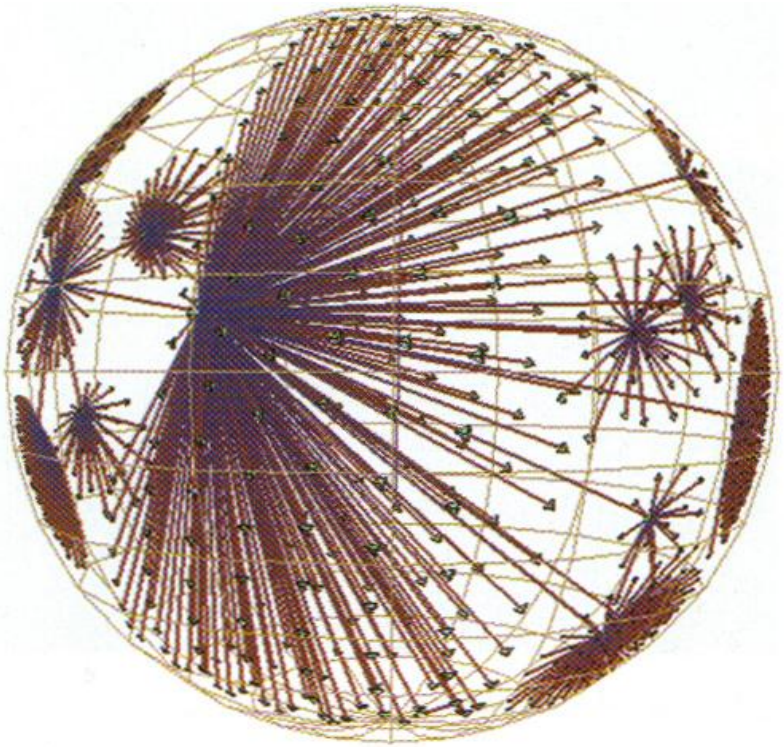



The Klein model for the hyperbolic plane. The line segment **AB** and **A'B'** have an equal length in the hyperbolic sense.

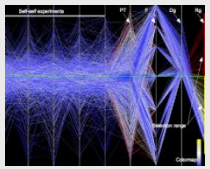


# 5.5 Exploration großer Graphen

- **Hyperbolischer Ansatz** legt nun Baum oder Graph in unendlich großen hyperbolischen Ebene aus und zeichnet das euklidische Modell



Hyperbolic view of a tree in 3D. (Courtesy of T. Munzner, Stanford University.)

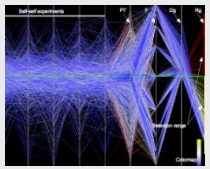




## Clustering

- Insbesondere, wenn **Graph zu groß** für Darstellung ist, kann man ihn **vereinfachen**
- Erfolgt in der Regel durch **Clustering der Knoten** mit folgenden Grundprinzipien:
  - **Structural-Clustering** - es wird nur aufgrund der **Struktur des Graphen** zusammengefasst
  - **Content-based Clustering** - es wird **Semantik** (insbesondere der Knoten) mit berücksichtigt

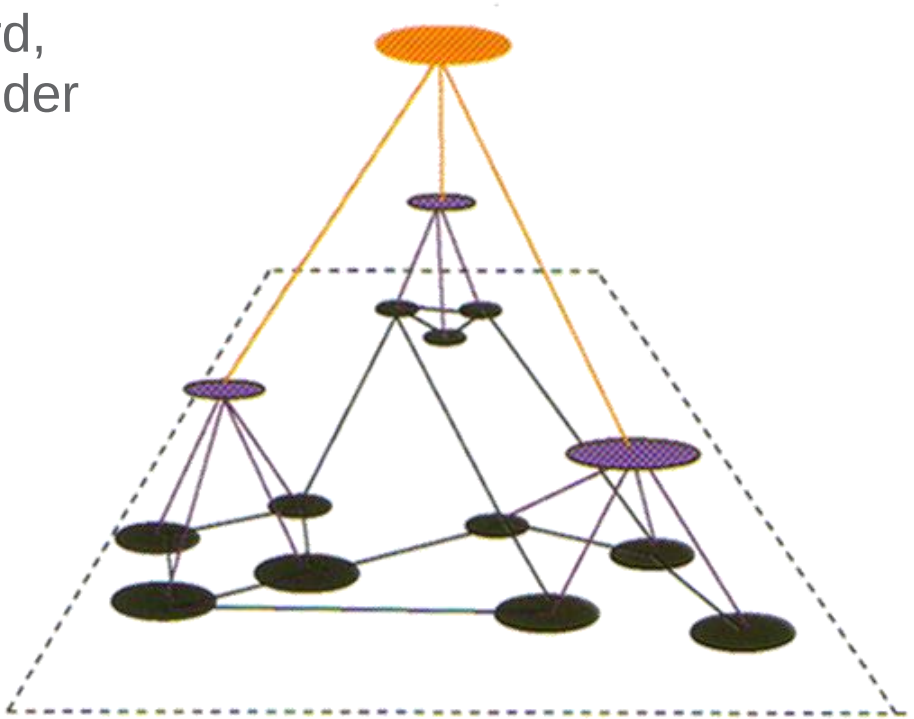
- Fast alle Verfahren basieren auf **Structural-Clustering**, da
  - es einfacher umzusetzen ist
  - Ansatz auf jeden Graphen unabhängig von Anwendungsdomäne angewendet werden kann



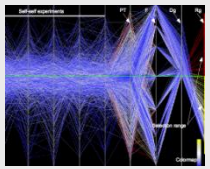


## Clustering, Forts.

- Es werden praktisch immer **disjunkte Cluster** erzeugt
- Aus Clustern wird neuer Graph erzeugt, indem
  - **Cluster zu Knoten werden** und
  - **Kante zwischen Clustern** gebildet wird, wenn eine Kante zwischen Elementen der Cluster existiert



A structure induced by hierarchical clustering. (Adapted from Eades and Feng [37].)

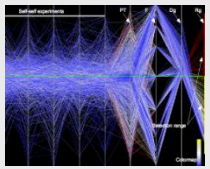




## Clustering, Forts.

- Für **Clustering** ist **Metrik der Knoten** nötig
- **Metriken** können strukturell oder inhaltsbasiert sein und ergeben so die beiden Typen des Clusterings.
- **Metrik von Furnas** mischt **Distanz von einem Knoten** zum anderen mit **gewünschten Detailgrad** für jeweiligen Bereich des Graphen

[Furnas, Generalized Fisheye Views, Human Factors in Computing Systems, CHI'86 Conference Proceedings, 1986, 16-23]

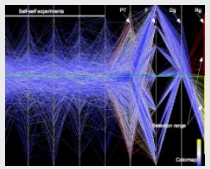




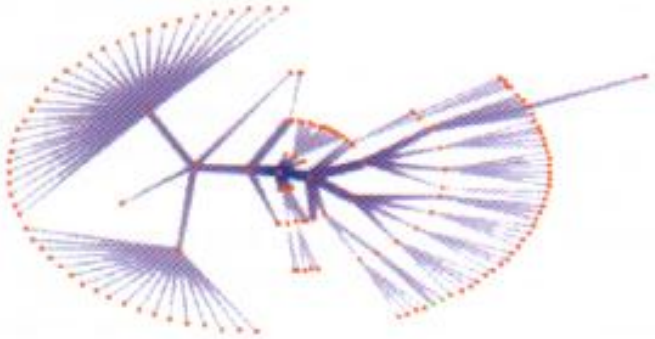
## Clustering, Forts.

- Man kann Knoten und Kanten auch einfach einen **Relevanzwert** geben
- Für Repräsentation gibt es drei verschiedene Ansätze
  - **Ghosting** - unwichtige Kanten und Knoten treten in **Hintergrund**
  - **Hiding** - unwichtige Elemente **weglassen**
  - **Grouping** - unwichtige Elemente **zusammenfassen**

[Kimelman, Leban, Roth, Zernik, Reduction of visual complexity in dynamic graphs, Proc. Symp. Graph Drawing GD'94, 1994]

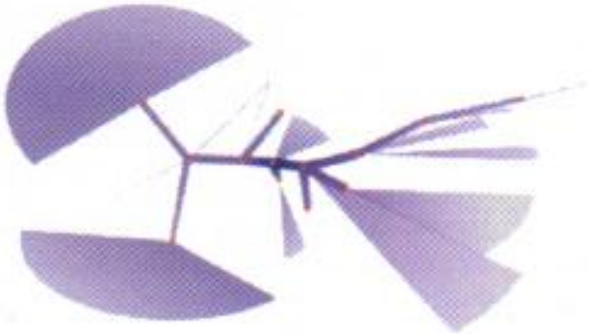


# 5.5 Exploration großer Graphen



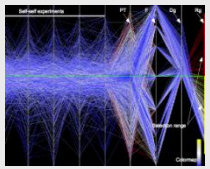
(a)

(b)



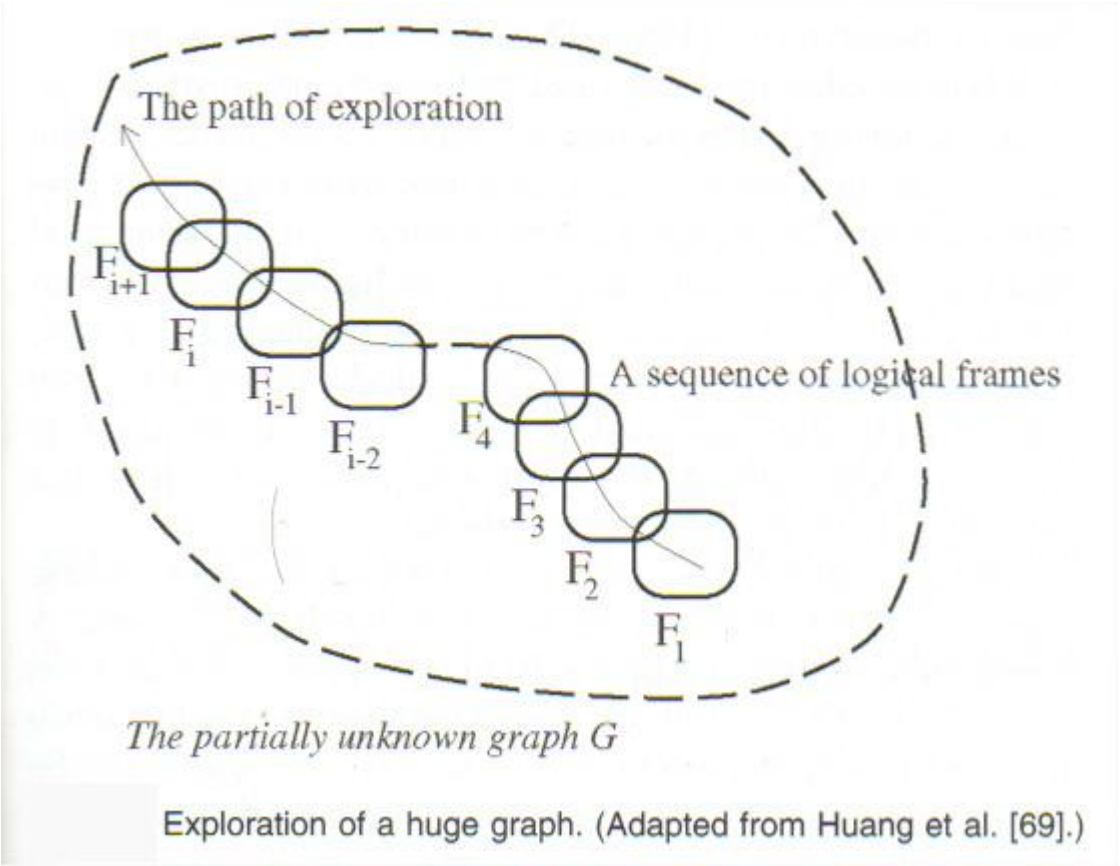
(c)

Fig. 22. Different schematic views of a tree: (a) ghosting, (b) hiding, and (c) grouping.

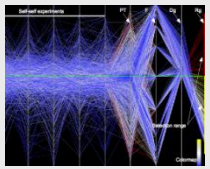


# 5.5 Exploration großer Graphen

- Große Graphen mit Hilfe einer **Fenstertechnik** untersuchen



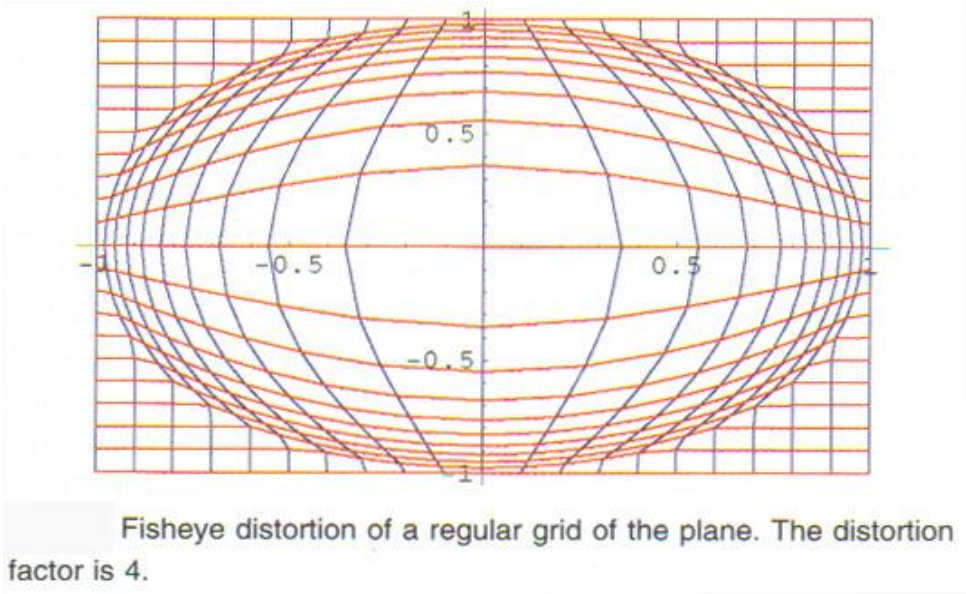

Exploration of a huge graph. (Adapted from Huang et al. [69].)



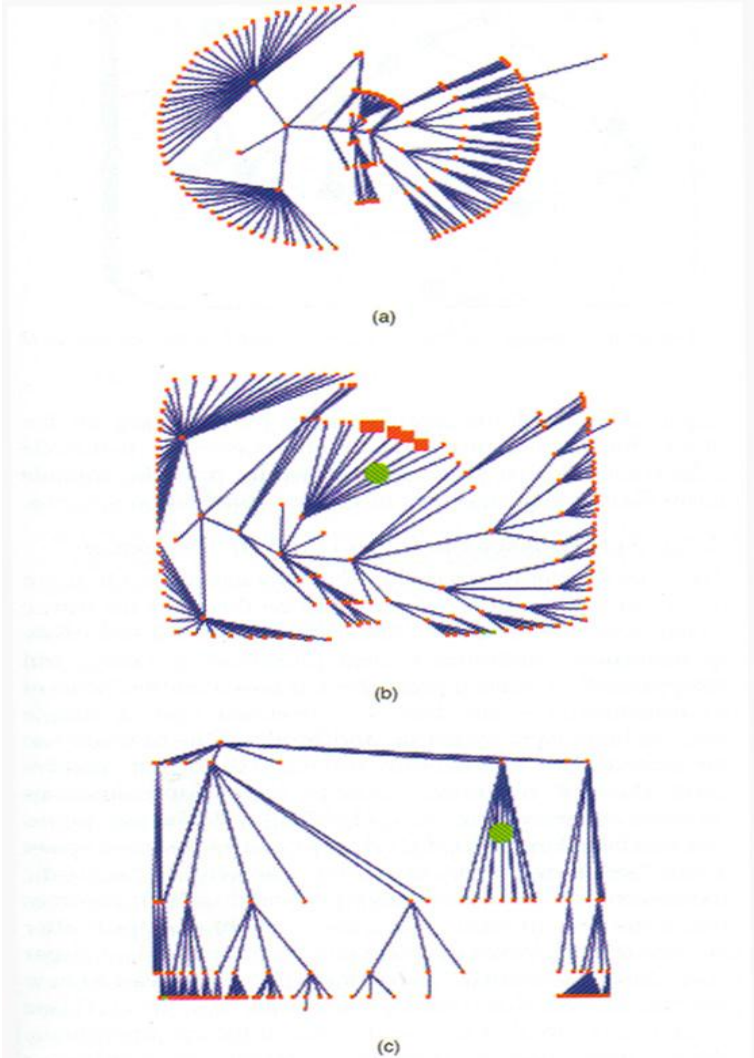


## Exploration

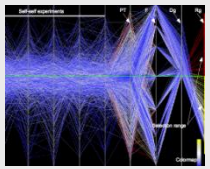
- Bei großen Graphen kann eine **Fischaugen-projektion** zur Untersuchung helfen




Fisheye distortion of a regular grid of the plane. The distortion factor is 4.




Fisheye distortion. (a) Represents the graph without the fisheye. (b) Uses polar fisheye, whereas (c) uses Cartesian fisheye with a different layout of the same graph. The green dots on (b) and (c) denote the focal points of the fisheye distortion. Note the extra edge-crossing on (b).

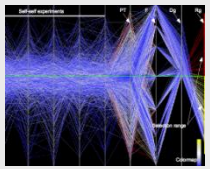




## Offene Fragen und Trends

- **Visualisierung sehr großer Graphen** mit mehreren Millionen oder gar Milliarden Knoten und Kanten (z.B. der Abstammungsbaum aller biologischen Arten, alle Webseiten, alle Festnetztelefone, das momentane Handynetz, Internet,...)
- Ansätze des Graphenzeichnens müssen mit **schneller Interaktion** und **effizienten Datenstrukturen** verknüpft werden

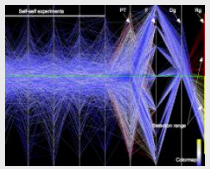
- Einige Graphentypen sind bisher weniger beachtet worden:
  **Hypergraphen**
  - Lassen sich als bipartite Graphen auffassen
    - Wenn für jede Hyperkante ein Zusatzknoten einfügt wird
    - Ist dann mit beteiligten Knoten verbunden und können dann gezeichnet werden.
  - Anwendungen (z. B. Stoffwechsel in der Biologie) erfordern jedoch besondere Erweiterungen, die Gegenstand aktueller Arbeiten sind.



# 5.6 Ausblick

## Offene Fragen und Trends

- Fragen offen trotz vieler Ansätze bei
  - Visualisierung **komplexer Ströme in Netzwerken**
  - **Übergang zu Prozessen**



# Literatur


- Herman, Melancon, Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. IEEE TVCG 6(1): 24-43, 2000
- G. Di Battista, P. Eades, R. Tomassia, I. G. Tollis. Graph Drawing. Prentice-Hall, Upper Saddle River, NJ, USA, 1999