PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Redes

Visualización de Información
IIC2026

Profesor: Denis Parra

Clase de hoy: Redes en el framework de Tamara

Arrange:

1. Organizar
2. Ordenar
3. Arreglar
4. Disponer
5. Concertar

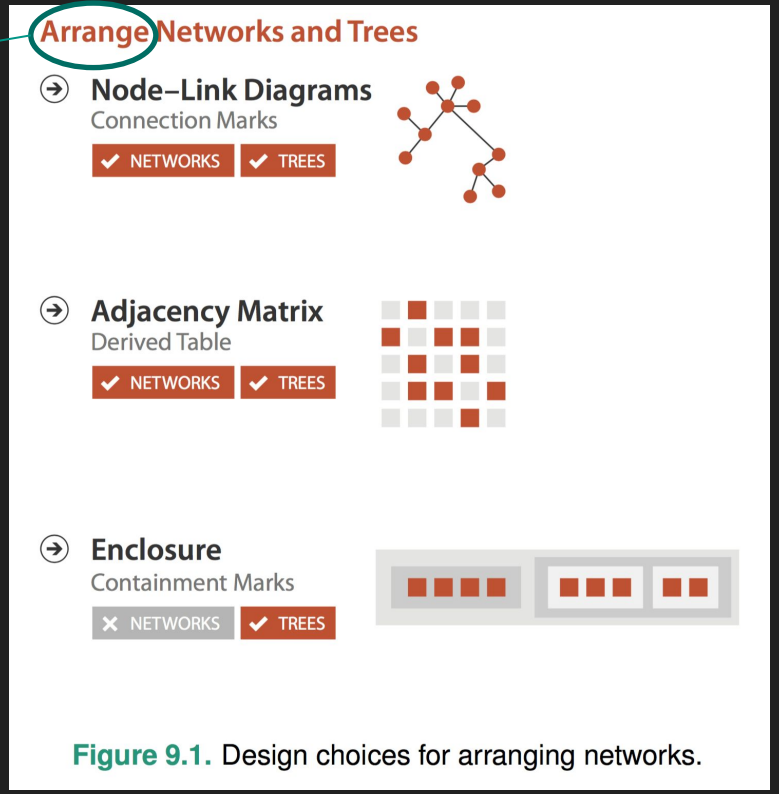


Figure 9.1. Design choices for arranging networks.

Codificaciones Visuales

- Diagrama nodo enlace: Canal de conexión muesta enlaces
- Diagrama/Vista matricial (relación de adyacencia)
- Diagrama/estructura de árbol: canal de contenimiento, enlaces muestran relaciones de jerarquía

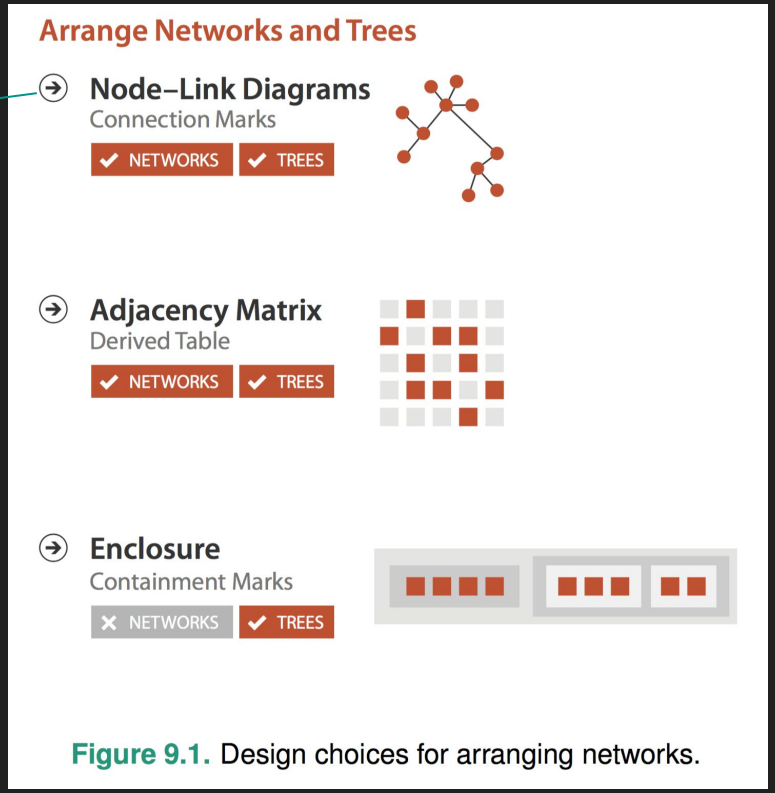


Figure 9.1. Design choices for arranging networks.

Codificaciones Visuales

- Diagrama nodo enlace: Canal de conexión muesta enlaces
- Diagrama/Vista matricial (relación de adyacencia)
- Diagrama/estructura de árbol: canal de contenimiento, enlaces muestran relaciones de jerarquía

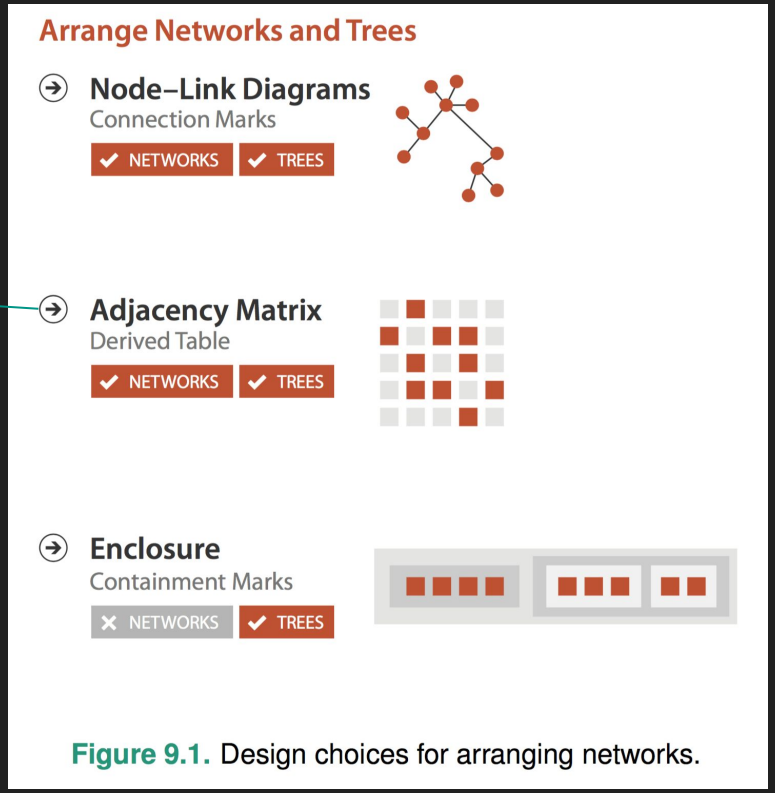


Figure 9.1. Design choices for arranging networks.

Codificaciones Visuales

- Diagrama nodo enlace: Canal de conexión muesta enlaces
- Diagrama/Vista matricial (relación de adyacencia)
- Diagrama/estructura de árbol: canal de contenimiento, enlaces muestran relaciones de jerarquía

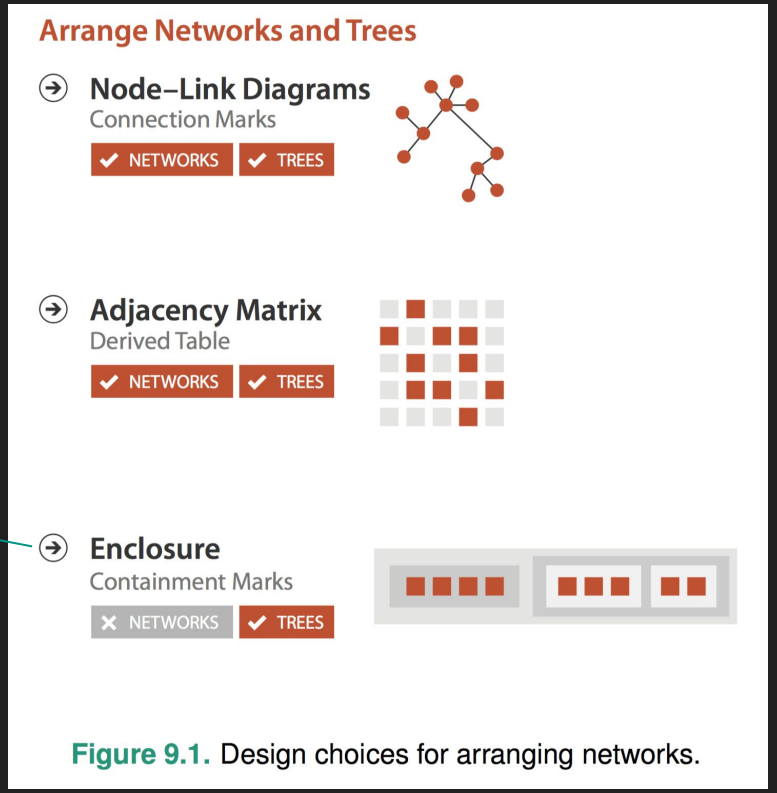


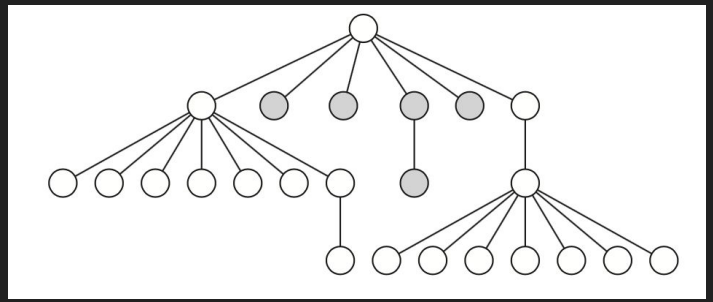
Figure 9.1. Design choices for arranging networks.

Canal de Conexión: Marcas de Enlace

- Los diagramas nodo-enlace son la codificación visual más común para datos de árboles y redes.
- Nodos son dibujados como marcas de punto
- Enlaces conectando nodos son dibujados como marcas de linea

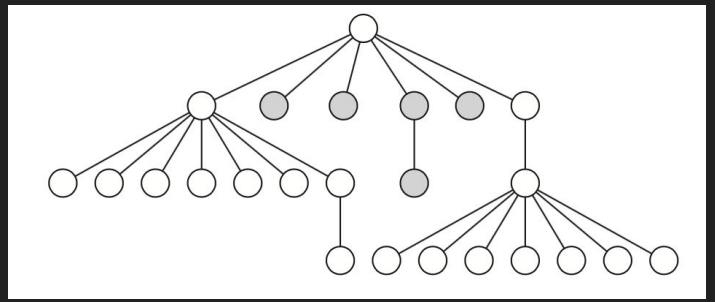
Canal de Conexión: Árboles

- El siguiente árbol presenta 24 nodos, con la raíz en el punto más alto y las hojas en las posiciones más bajas.
- Aparte del canal de conexión, ¿qué otro canal utiliza?



Canal de Conexión: Árboles

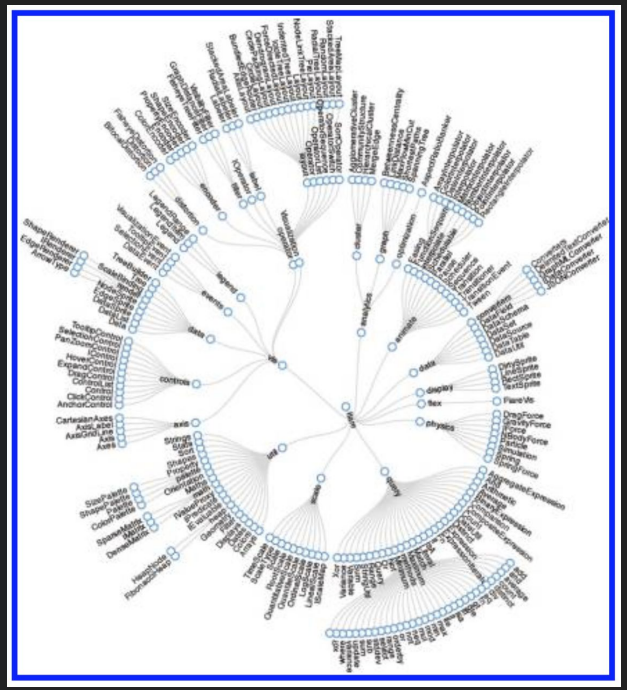
- El siguiente árbol presenta 24 nodos, con la raíz en el punto más alto y las hojas en las posiciones más bajas.
- Aparte del canal de conexión, ¿qué otro canal utiliza?



El canal espacial vertical representa la profundidad del árbol

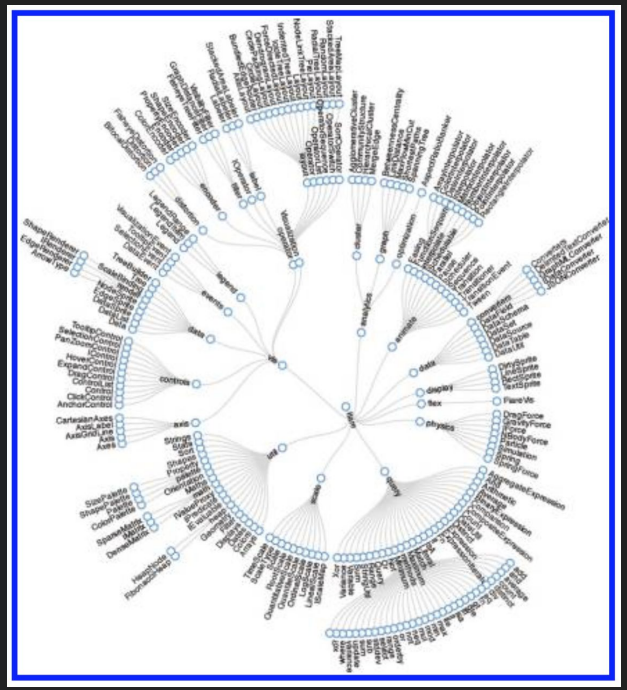
Canal de Conexión: Árboles

- El siguiente árbol presenta unos cientos de nodos en un spline radial layout.
- Codificación visual es radial en lugar de rectilínea.
- ¿ Qué canal se usa para representar profundidad?



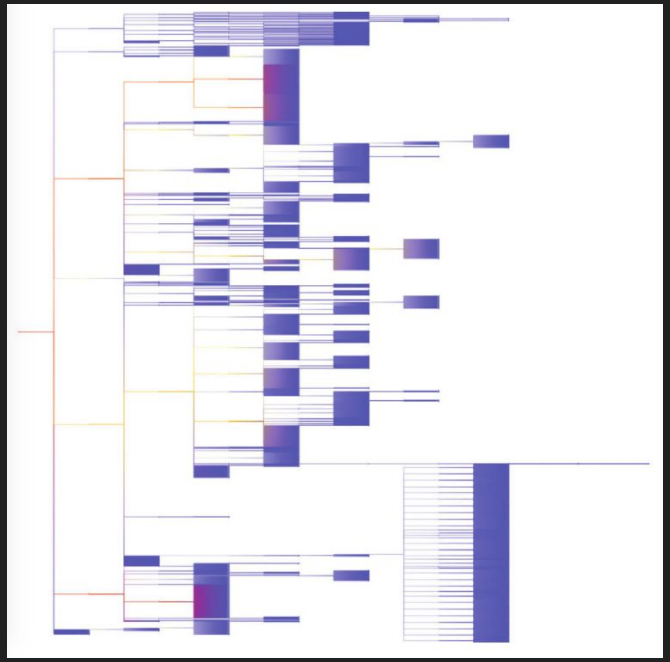
Canal de Conexión: Árboles

- El siguiente árbol presenta unos cientos de nodos en un spline radial layout.
- Codificación visual es radial en lugar de rectilínea.
- ¿ Qué canal se usa para representar profundidad? Canal espacial, distancia respecto a la raíz.

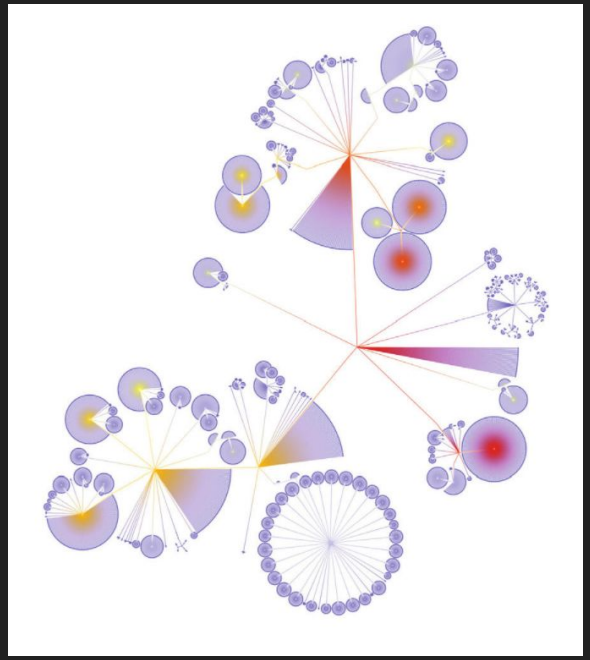


Canal de Conexión: Árboles – 5161 nodos

Layout rectangular horizontal

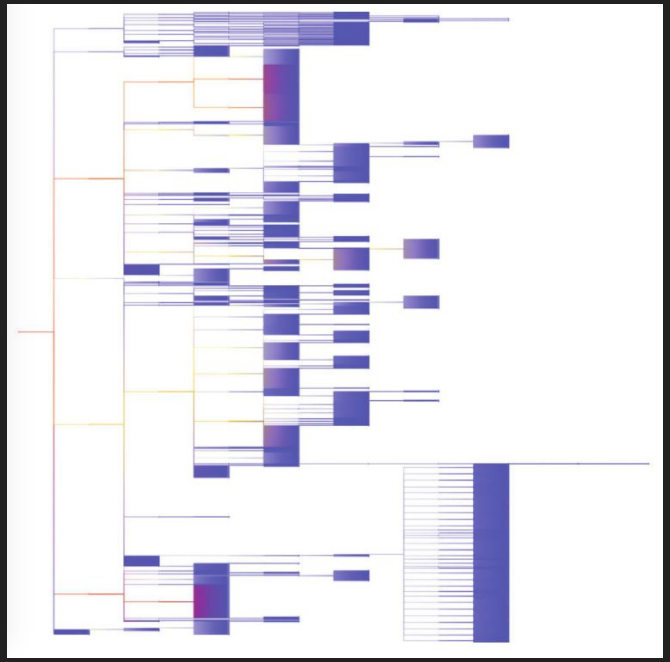


BubbleTree



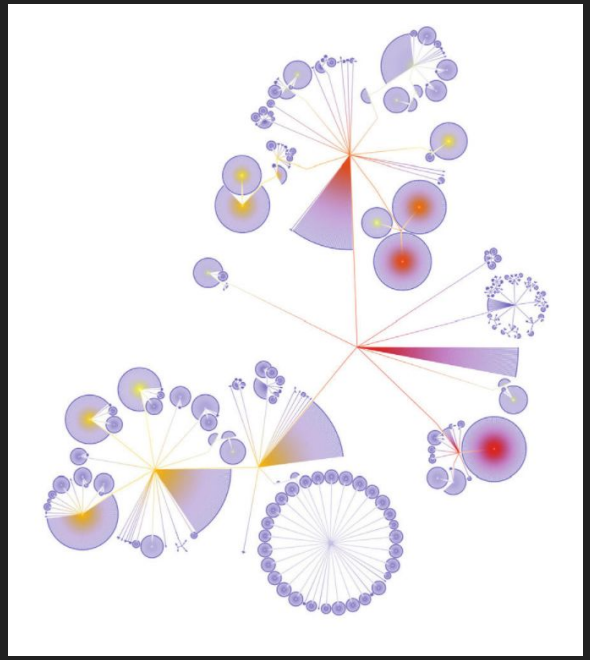
Canal de Conexión: Árboles – 5161 nodos

Layout rectangular horizontal



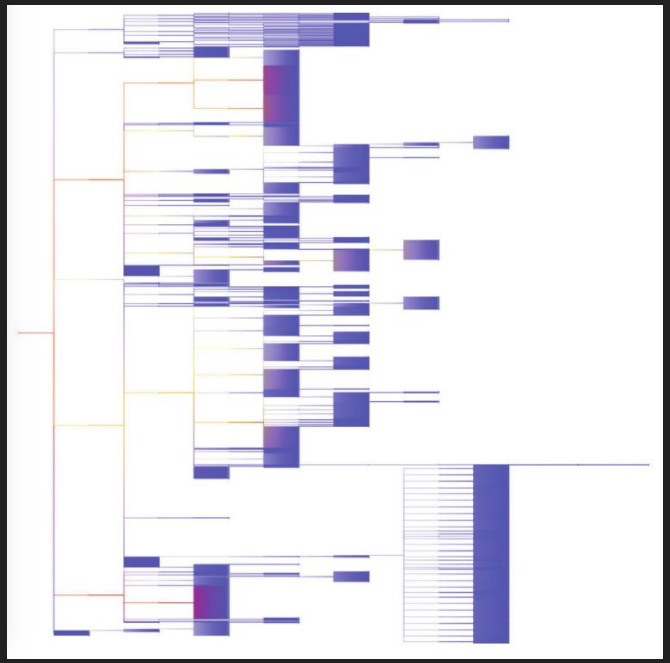
- Líneas representan enlaces coloreados en base a Strahler centrality

BubbleTree (Grivet et al, 06)



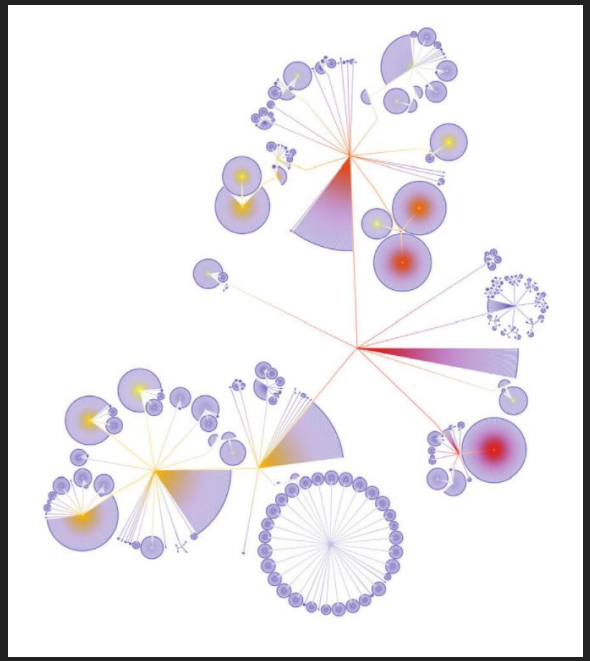
Canal de Conexión: Árboles – 5161 nodos

Layout rectangular horizontal



- Subárboles son dibujados como círculos completos en lugar de arcos parciales.

BubbleTree (Grivet et al, 06)



Canal de Conexión: Redes

- Los diagramas nodo-enlace también son comunes para representar redes.
- El número de saltos (hops) en una ruta entre dos nodos es una métrica de distancia. Métrica discreta (no continua, como en un plano 2D)
- Diagramas nodo-enlace encajan bien en tareas relacionadas con entender la topología de una red: conexiones directas o indirectas entre nodos.
- Ejemplos: encontrar todos los caminos entre dos nodos, encontrar todos los nodos adyacentes en 1 salto, encontrar los nodos puente de una red.

Redes: Posicionamiento Force-directed I

- El force-directed placement es una de los idioms más comunes para marcas de conexión.
- Variantes: posicionar simulando fuerzas físicas que empujan los nodos entre sí, mientras los enlaces actúan como resortes para acercar sus endpoints.
- Force-directed placement también se conoce como Spring embedding, minimización de energía u optimización no lineal.

Redes: Posicionamiento Force-directed II

- Los algoritmos de force-directed placement usualmente parten ubicando los nodos de forma aleatoria dentro de una región espacial, y luego refinan iterativamente sus posiciones de acuerdo a simulación de fuerzas.
- Ventajas: relativamente fácil de implementar, fácil de explicar la intuición detrás del algoritmo.
- Desventajas: la posición espacial no codifica valores de atributos, ya que intentan simplemente disminuir cruce de enlaces y sobreposición de nodos.
  - Posicionamiento es no determinístico: no siempre permite explotar memoria espacial.
  - Escalabilidad: Computacionalmente y Visualmente (hairballs)

Algoritmos de posicionamiento Force-Directed

- El primero de estos métodos fue propuesto en 1984 por Peter Eades

Eades, P. (1984). A heuristic for graph drawing. Congressus numerantium, 42, 149-160.

- Idea Principal: << Dado un grafo, reemplazamos los vertices por aros de acero y los enlaces por resortes para formar un Sistema mecánico >>
- <<Los vertices son ubicados en algún layout inicial y luego dejamos de las fuerzas de los resortes muevan el sistema a un estado de energía minima>>

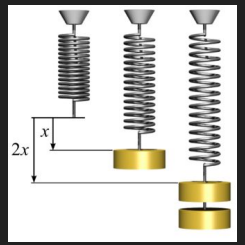
Eades (1984)

- El algoritmo se ve más o menos así:

```
Algoritmo Resorte (G: grafo)
    ubicar vertices de G en ubicaciones aleatorias;
    repetir M veces
        calcular la fuerza sobre cada vértice;
        mover el vértice c4 * (fuerza en el vértice)
    dibujar
```

Eades (1984)

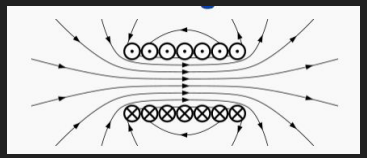
- Cómo calculary las fuerzas entre los nodos (vértices)
  - Entre los vertices conectados por enlaces, usar atracción vía ley de Hooke



$$F_s = kx$$

Fs: Fuerza sobre el resorte (**s**pring)
k: constante del resorte (stiffness)
x : desplazamiento

  - Entre los vertices no conectados, usar repulsion vía ley de Coulomb



$$F = k_e \frac{q_1 q_2}{r^2},$$

F: Fuerza de repulsion/atracción
ke: constante de Coulomb
q1 y q2 : magnitude de las cargas
r: distancia entre las cargas

Springy (https://github.com/dhotson/springy)

What is Springy?

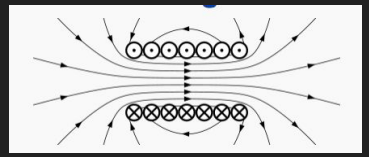
Springy is a force directed graph layout algorithm.

So what does this "force directed" stuff mean anyway? *Excellent question!*

It means that springy uses some real world physics to try and figure out how to show a network graph in a way that looks good.

```
Layout.ForceDirected.prototype.tick = function(timestep) {
    this.applyCoulombsLaw();
    this.applyHookesLaw();
    this.attractToCentre();
    this.updateVelocity(timestep);
    this.updatePosition(timestep);
};
```

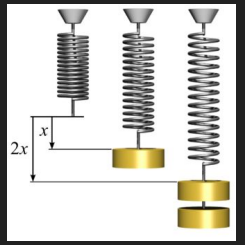
Springy Coulomb's Law



$$F = k_e \frac{q_1 q_2}{r^2},$$

```
// Physics stuff
Layout.ForceDirected.prototype.applyCoulombsLaw = function() {
    this.eachNode(function(n1, point1) {
        this.eachNode(function(n2, point2) {
            if (point1 !== point2)
            {
                var d = point1.p.subtract(point2.p);
                var distance = d.magnitude() + 0.1; // avoid massive forces at small distances (and divide by zero
                var direction = d.normalise();

                // apply force to each end point
                point1.applyForce(direction.multiply(this.repulsion).divide(distance * distance * 0.5));
                point2.applyForce(direction.multiply(this.repulsion).divide(distance * distance * -0.5));
            }
        });
    });
};
```

Springy Hooke's Law



$$F_s = kx$$

```
Layout.ForceDirected.prototype.applyHookesLaw = function() {
    this.eachSpring(function(spring){
        var d = spring.point2.p.subtract(spring.point1.p); // the direction of the spring
        var displacement = spring.length - d.magnitude();
        var direction = d.normalise();

        // apply force to each end point
        spring.point1.applyForce(direction.multiply(spring.k * displacement * -0.5));
        spring.point2.applyForce(direction.multiply(spring.k * displacement * 0.5));
    });
};
```

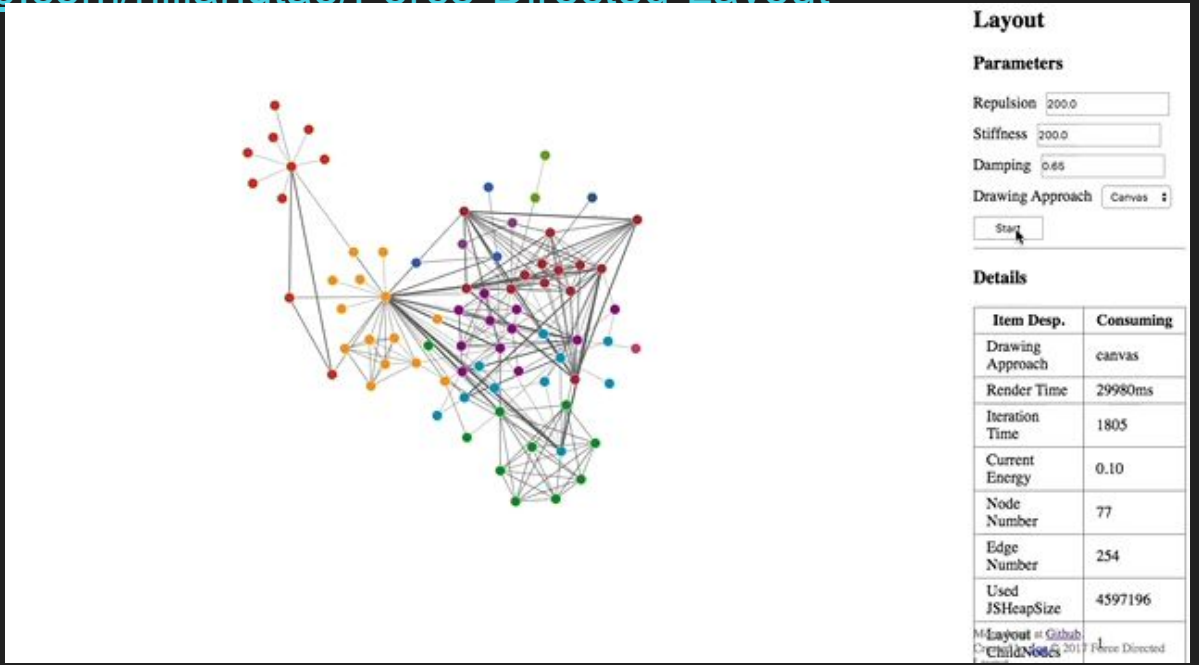
Springy: Energía del Sistema

```
Layout.ForceDirected.prototype.start = function(render, onRenderStop, onRenderStart) {
    var t = this;

    if (this._started) return;
    this._started = true;
    this._stop = false;

    if (onRenderStart !== undefined) { onRenderStart(); }

    Springy.requestAnimationFrame(function step() {
        t.tick(0.03);

        if (render !== undefined) {
            render();
        }

        // stop simulation when energy of the system goes below a threshold
        if (t._stop || t.totalEnergy() < t.minEnergyThreshold) {
            t._started = false;
            if (onRenderStop !== undefined) { onRenderStop(); }
        } else {
            Springy.requestAnimationFrame(step);
        }
    });
};
```

```
// Calculate the total kinetic energy of the system
Layout.ForceDirected.prototype.totalEnergy = function(timestep) {
    var energy = 0.0;
    this.eachNode(function(node, point) {
        var speed = point.v.magnitude();
        energy += 0.5 * point.m * speed * speed;
    });

    return energy;
};
```

Demo

- Otra implementación pedagógica: https://github.com/hijiangtao/Force-Directed-Layout



Fruchtermann & Reingold (1991)

- Agregan al modelo de Eades “distribución uniforme de vertices” donde fuerzas atractivas y repulsivas se dan por:

$$f_a(d) = d^2/k, \qquad f_r(d) = -k^2/d,$$

- La distancia idea entre vertices k se define como

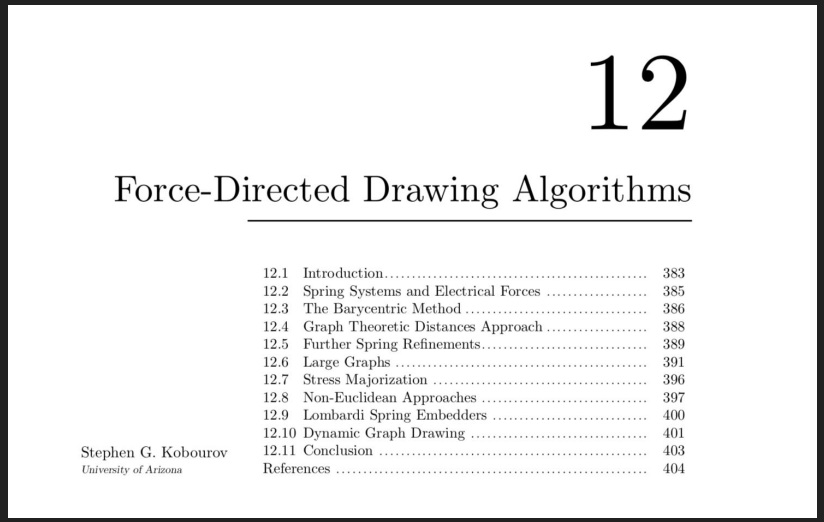
$$k = C\sqrt{\frac{area}{number\ of\ vertices}}.$$

Fruchtermann & Reingold (1991)

- Adicionalmente, el algoritmo de Fructerman y Reingold añade la noción de ***temperatura***.
- “the temperature could start at an initial value (say one tenth the width of the frame) and decay to 0 in an inverse linear fashion.”
- La temperatura controla el desplazamiento de los vértices de forma que a medida que el layout mejora, los ajustes se hacen más pequeños. El uso de temperature es un caso especial de una técnica llamada ***simulated annealing.***

Más detalles

- Capítulo de libro “Force-Directed Drawing Algorithms” de Stephen G. Kobourov
- http://cs.brown.edu/people/rtamassi/gdhandbook/chapters/force-directed.pdf



12

Force-Directed Drawing Algorithms

Stephen G. Kobourov
University of Arizona

Redes: Posicionamiento Force-directed III

- Interpretación debe hacerse con cuidado: grupo de nodos interconectados puede indicar un cluster, pero en ocasiones, nodos espacialmente cercanos a varios hops de distancia pueden ser sólo artefacto del algoritmo.

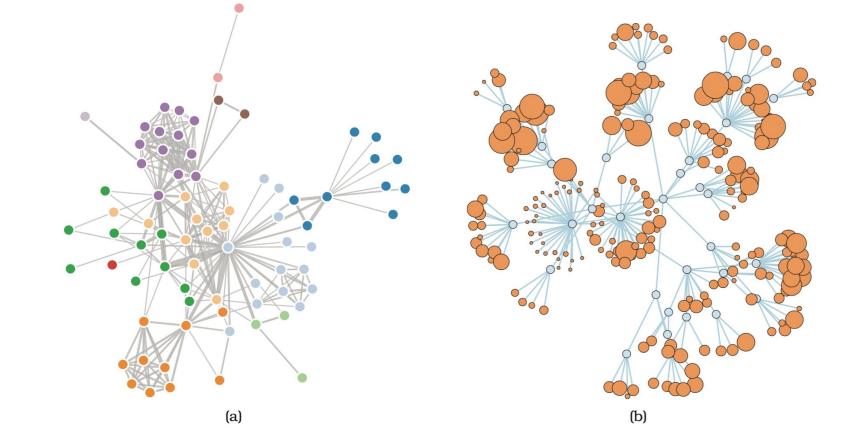


Figure 9.4. Node–link layouts of small networks. (a) Force-directed placement of small network of 75 nodes, with size coding for link attributes. (b) Larger network, with size coding for node attributes. From http://bl.ocks.org/mbostock/4062045 and http://bl.ocks.org/1062288.

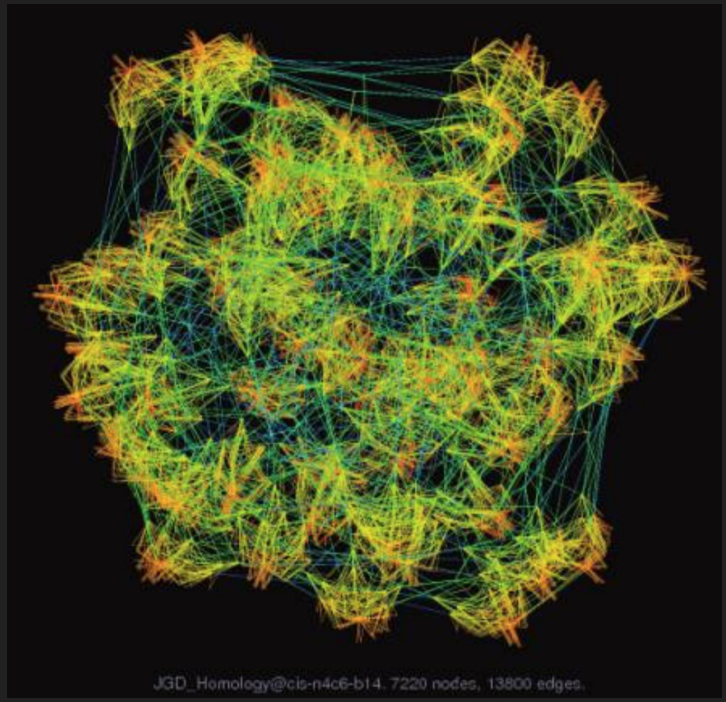
Redes: Posicionamiento Force-directed IV

- Posicionamiento es no determinístico: no siempre permite explotar memoria espacial.
- Escalabilidad: Computacionalmente y Visualmente (hairballs)

Idiom	**Force-Directed Placement**
What: Data	Network.
How: Encode	Point marks for nodes, connection marks for links.
Why: Tasks	Explore topology, locate paths.
Scale	Nodes: dozens/hundreds. Links: hundreds. Node/link density: $L < 4N$

Redes: Opciones para Force-Directed

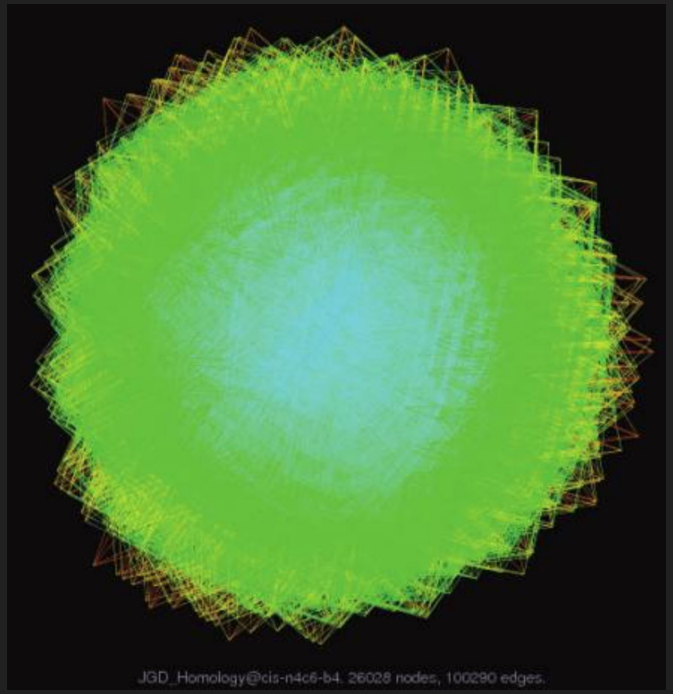
- Varios modelos recientes utilizan idiom de redes multinivel, donde la red original se “aumenta” con una jerarquía de cluster derivada.
- Se pueden usar algoritmos de detección de comunidad como Girvan-Newman, affinity propagation o maximum-modularity.



SFDP (Hu, 05) 7.220 nodos y 13.800 enlaces

Redes: Opciones para Force-Directed

- Varios modelos recientes utilizar idiom de redes multinivel, donde la red original se “aumenta” con una jerarquía de cluster derivada.
- Se pueden usar algoritmos de detección de comunidad como Girvan-Newman, affinity propagation o maximum-modularity.



SFDP (Hu, 14) 26.028 nodos y 100.290 enlaces

Redes: Opciones para Force-Directed

Idiom	**Multilevel Force-Directed Placement (sfdp)**
What: Data	Network.
What: Derived	Cluster hierarchy atop original network.
What: Encode	Point marks for nodes, connection marks for links.
Why: Tasks	Explore topology, locate paths and clusters.
Scale	Nodes: 1000–10,000. Links: 1000–10,000. Node/link density: $L < 4N$.

Matrix View

- Mayor Capacidad de Escalabilidad de el diagrama nodo-enlace.
- Vista ideal para ciertas tareas de red:
  - Encontrar un nodo dado una etiqueta, estimar cantidad de nodos en una red
  - Posicionamiento estable, predecible, con posibilidad de reordenar

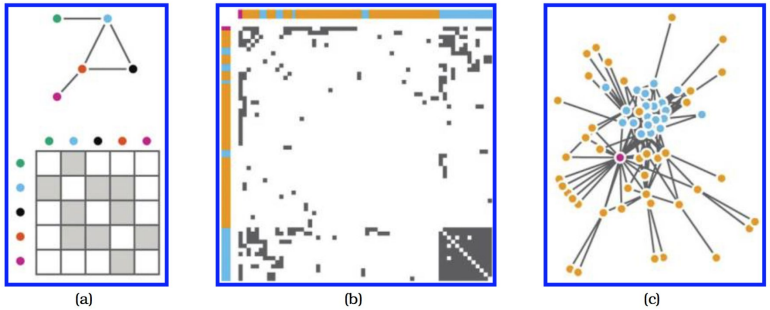


Figure 9.6. Comparing node–link matrix and matrix views of a network. (a) Node–link and matrix views of small network. (b) Matrix view of larger network. (c) Node–link view of larger network. From [Gehlenborg and Wong 12, Figures 1 and 2].

Idiom	**Adjacency Matrix View**
What: Data	Network.
What: Derived	Table: network nodes as keys, link status between two nodes as values.
How: Encode	Area marks in 2D matrix alignment.
Scale	Nodes: 1000. Links: one milllion.

Connection View vs. Matrix View

- Para redes pequeñas, el diagrama nodo-enlace es óptimo pues es intuitivo y muchas tareas requieren revisar enlaces, rutas y topología.
- Para redes con gran densidad de enlaces, el diagrama nodo-enlace es claramente sub-óptimo -> hairball (bola de pelo)
- La vista de matriz es poco familiar, especialmente para tareas relacionadas con estudiar topología de la red: se requiere entrenamiento para interpretar la vista de red.

Connection View with Matrix View: Bicliques

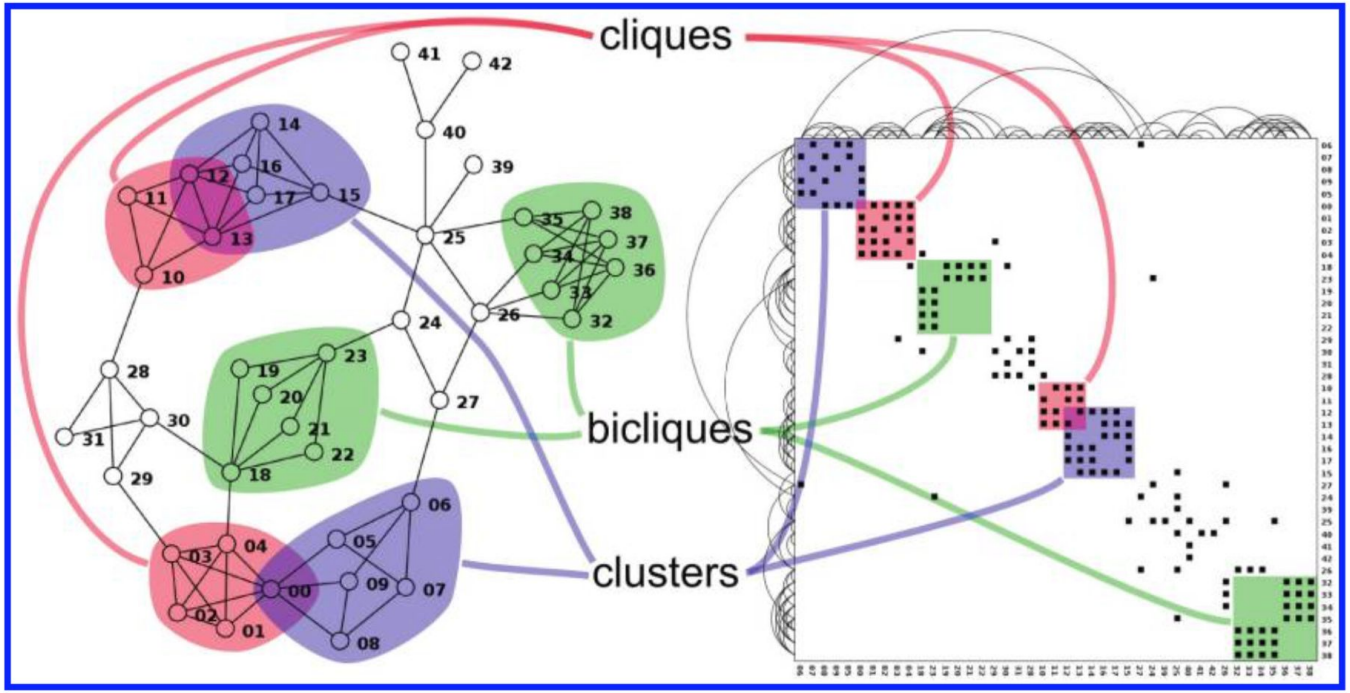


Figure 9.7. Characteristic patterns in matrix views and node–link views: both can show cliques and clusters clearly. From [McGuffin 12, Figure 6].

Contenimiento: TreeMaps

- Vista óptima para relación de jerarquía en lugar de conexiones.

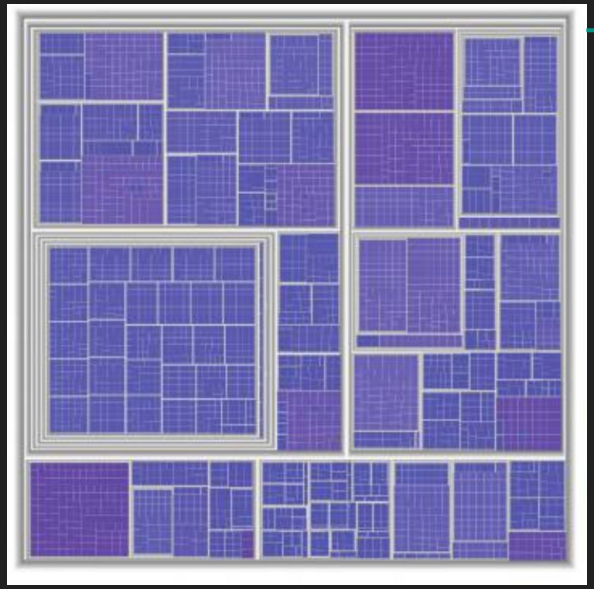


Figure 9.8. Treemap layout showing hierarchical structure with containment rather than connection, in contrast to the node–link diagrams of the same 5161-node tree in Figure 9.3.

Idiom	**Treemaps**
What: Data	Tree.
How: Encode	Area marks and containment, with rectilinear layout.
Why: Tasks	Query attributes at leaf nodes.
Scale	Leaf nodes: one million. Links: one million.

Contenimiento: TreeMaps II

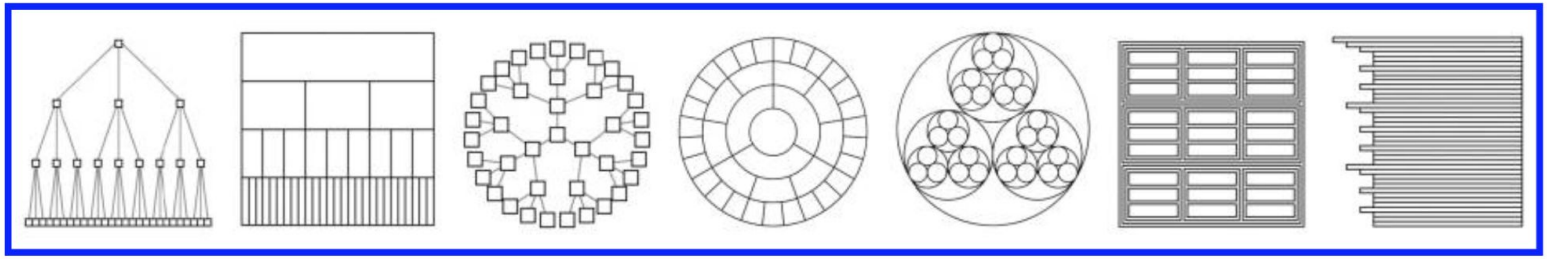


Figure 9.9. Seven visual encoding idioms showing the same tree dataset, using different combinations of visual channels. (a) Rectilinear vertical node–link, using connection to show link relationships, with vertical spatial position showing tree depth and horizontal spatial position showing sibling order. (b) Icicle, with vertical spatial position and size showing tree depth, and horizontal spatial position showing link relationships and sibling order. (c) Radial node–link, using connection to show link relationships, with radial depth spatial position showing tree depth and radial angular position showing sibling order. (d) Concentric circles, with radial depth spatial position and size showing tree depth and radial angular spatial position showing link relationships and sibling order. (e) Nested circles, using radial containment, with nesting level and size showing tree depth. (f) Treemap, using rectilinear containment, with nesting level and size showing tree depth. (g) Indented outline, with horizontal spatial position showing tree depth and link relationships and vertical spatial position showing sibling order. From [McGuffin and Robert 10, Figure 1].

Ejemplo: GrouseFlocks

- Ejemplo de Contenimiento combinando red y árbol: compound network

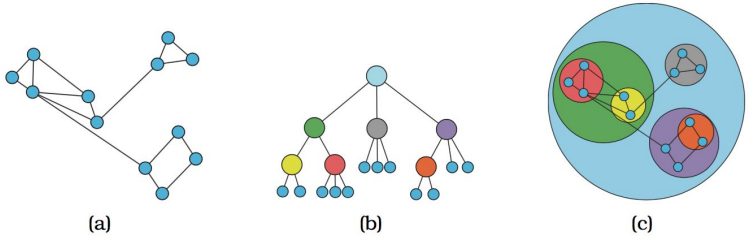


Figure 9.10. GrouseFlocks uses containment to show graph hierarchy structure. (a) Original graph. (b) Cluster hierarchy built atop the graph, shown with a node–link layout. (c) Network encoded using connection, with hierarchy encoded using containment. From [Archambault et al. 08, Figure 3].

System	**GrouseFlocks**
What: Data	Network.
What: Derived	Cluster hierarchy atop original network.
What: Encode	Connection marks for original network, containment marks for cluster hierarchy.

Referencias y Lecturas adicionales

- Revisar Tree Drawings Space: http://treevis.net

Práctico

- http://dparra.sitios.ing.uc.cl/classes/infovis-2019-2/NetworkViz-tutorial-2019.pdf