Layout de Grafos en 3D con Modelos de Fuerzas Intercambiables: FR, LinLog y Spring–Electrical sobre OpenGL Interactivo

Sebastián Alzate
sebastian.alzate@upr.edu
Universidad de Puerto Rico, Recinto de Mayagüez

Resumen

Este trabajo presenta un sistema interactivo para el layout de grafos en 3D basado en modelos de fuerzas clásicos y variantes modernas. El sistema implementa tres modelos conmutables en tiempo real: Fruchterman–Reingold (FR), LinLog y un esquema Spring–Electrical (SE) con parámetros interpretables. La aplicación está escrita en C++ con OpenGL y GLUT, e integra un *HUD* con métricas (FPS, desplazamiento promedio, temperatura), suavizado temporal de posiciones y coloración por comunidades vía k-means++. El programa acepta grafos desde archivos CSV genéricos y se valida sobre el grafo social clásico del Karate Club de Zachary, además de un grafo sintético de referencia. Los resultados muestran que FR ofrece un *baseline* compacto y estable, LinLog mejora la separación de comunidades y SE permite afinar el espaciado y resaltar puentes mediante el ajuste de los parámetros físicos. El sistema mantiene tasas de refresco interactivas para grafos pequeños y medianos, y sirve como herramienta didáctica para explorar la relación entre energía, fuerzas y estructura de grafos.

Palabras clave: layout de grafos, modelos de fuerzas, visualización 3D, Fruchterman–Reingold, LinLog, Spring–Electrical, OpenGL.

1. Introducción

La visualización de grafos es una herramienta central en análisis de redes sociales, bioinformática, sistemas de recomendación y muchas otras áreas. En la práctica, la mayoría de layouts utilizados en herramientas estándar se basan en variantes de modelos de fuerzas en 2D [1,2]. Sin embargo, muchos grafos reales tienden a ser densos, con comunidades fuertemente conectadas, lo que da lugar al típico "hairball" difícil de interpretar.

Trabajar en 3D abre una dimensión adicional para separar módulos, reducir cruces de aristas y explorar la estructura del grafo mediante rotaciones, zoom y parámetros físicos controlables. Por otro lado, la simulación en 3D es más costosa y sensible a inestabilidades numéricas, lo que plantea retos de rendimiento y de diseño de interacción.

Este proyecto desarrolla un sistema compacto en C++ y OpenGL que implementa tres modelos de fuerzas conmutables (FR, LinLog y Spring–Electrical) para layouts en 3D, con un foco explícito en:

- **Interactividad**: actualización en tiempo real con 30–60 FPS para grafos pequeños/medianos.
- **Control visual**: pocos parámetros, pero interpretables (temperatura, cooling, constantes de resortes).
- **Lectura estructural**: separación de comunidades, identificación de hubs y puentes.

1.1. Aportes principales

Los aportes concretos de este trabajo son:

- Implementación de un layout en 3D con tres modelos de fuerzas intercambiables (FR, LinLog y SE) sobre un único núcleo de simulación.
- Un *HUD* minimalista que expone métricas (FPS, desplazamiento promedio, parámetros físicos) para entender el comportamiento del sistema.
- Un módulo de lectura de grafos desde CSV genérico que maneja encabezados, etiquetas arbitrarias, lazos y aristas duplicadas.
- Una coloración por comunidades basada en k-means++ sobre las posiciones actuales, pensada para lectura rápida en demos interactivas.
- Un caso de estudio sobre el grafo del Karate Club de Zachary, comparando los tres modelos en términos de separación de facciones y resaltado de puentes.

2. Trabajo relacionado

Los algoritmos de layout por fuerzas tienen una larga historia en visualización de grafos. Kamada y Kawai propusieron un enfoque basado en distancias geodésicas y resortes con longitud ideal [3]. Fruchterman y Reingold introdujeron un modelo más simple y eficiente, con fuerzas de atracción y repulsión definidas explícitamente y un esquema de *cooling* para controlar la convergencia [1].

Noack propuso la familia de modelos LinLog, donde la energía está diseñada para alinear los mínimos con particiones de alta modularidad, lo que produce layouts que separan mejor las comunidades [2]. Existen también variantes spring–electrical que combinan resortes lineales en aristas con repulsión coulombiana entre to-

dos los pares de nodos [4].

En cuanto a visualización en 3D, varios trabajos han extendido estos modelos a espacios tridimensionales, con énfasis en exploración interactiva y layouts para grafos densos. Sin embargo, muchas implementaciones quedan embebidas en herramientas grandes. En este proyecto se busca una versión mínima pero didáctica, donde el código sea lo bastante compacto como para ser estudiado en un curso de visualización.

Finalmente, el grafo del Karate Club de Zachary [5] es un benchmark clásico para métodos de detección de comunidades y layouts por fuerzas, ya que contiene dos facciones bien definidas y algunos nodos puente entre ellas.

3. Modelos de fuerzas y dinámica

Sea un grafo no dirigido $G = (V, E)$ con $|V| = N$ nodos. Cada nodo i tiene una posición $\mathbf{p}_i \in \mathbb{R}^3$, y el sistema define fuerzas de repulsión global y atracción en aristas. Denotamos $d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$.

En todos los modelos, el sistema calcula una fuerza neta $\mathbf{D}_i$ por nodo y actualiza las posiciones con un paso limitado por una "temperatura" T:

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \min\Big(1, \frac{T}{\|\mathbf{D}_i\|}\Big)\,\mathbf{D}_i.$$

Además, se aplica un resorte radial suave hacia una esfera de radio r_0 y se recentra el grafo en el origen para eliminar el *drift* debido a traslaciones.

3.1. Fruchterman–Reingold (FR)

En FR se define una escala típica

$$k = \frac{2}{\sqrt[3]{N}}, \tag{1}$$

y las fuerzas se modelan como:

$$F_{\text{rep}}(d) \propto \frac{k^2}{d}, \tag{2}$$

$$F_{\text{att}}(d) \propto \frac{d^2}{k}. \tag{3}$$

En la implementación, la repulsión se aplica entre todos los pares (i, j) y la atracción solamente en las aristas $(i, j) \in E$. Este modelo tiende a producir layouts compactos, con una escala aproximadamente uniforme y buena legibilidad para grafos pequeños y medianos.

3.2. LinLog

El modelo LinLog se basa en una energía cuyo mínimo favorece la agrupación de vértices densamente conectados [2]. Una forma típica es:

$$E(P) = \sum_{(i,j)\in E} w_{ij}\,\log d_{ij} + \alpha \sum_{i<j} \log d_{ij},$$

donde el primer término agrupa vértices conectados y el segundo promueve repulsión global.

En este proyecto se mantiene la repulsión de FR (para aprovechar la misma escala) y se modifica la atracción para que crezca de manera logarítmica:

$$F_{\text{att}}(d) \propto \frac{1}{k}\,\log(1 + d).$$

El resultado práctico es que bloques densos tienden a permanecer compactos, mientras que los puentes y conexiones intermodulares generan separaciones más marcadas entre comunidades.

3.3. Spring–Electrical (SE)

El modelo spring–electrical combina resortes lineales en las aristas con repulsión coulombiana entre todos los nodos [4]. Se definen tres parámetros interpretables:

- C_r: intensidad de la repulsión $1/d^2$.
- C_a: rigidez del resorte lineal.
- L: longitud ideal de cada arista.

Las fuerzas se modelan como:

$$F_{\text{rep}}(d) \propto \frac{C_r}{d^2}, \tag{4}$$

$$F_{\text{att}}(d) = C_a(d - L). \tag{5}$$

En la implementación, al entrar al modo SE se inicializa L en función de k y se restringe (C_r, C_a, L) a rangos razonables para evitar explosiones numéricas. El usuario puede ajustar estos parámetros desde el teclado, lo que permite abrir o compactar el layout sin cambiar la topología.

3.4. Temperatura, *cooling* y *reheat*

La temperatura T controla el tamaño máximo del paso por iteración. El sistema aplica un *cooling* exponencial cuando está activado:

$$T \leftarrow \beta T, \quad \beta < 1,$$

y dispone de un mecanismo de *reheat* que:

- eleva T tras cambios bruscos (p. ej., añadir aristas o cambiar de modelo de fuerzas),
- ejecuta varias iteraciones con cooling desactivado para evitar quedar atrapado en malos mínimos locales,
- restablece el suavizado temporal para evitar "fantasmas" en la animación.

4. Carga de datos, coloración y HUD

4.1. Lectura robusta de grafos

El sistema acepta un archivo CSV de aristas, especificado como argumento de línea de comandos. El loader:

- admite separadores `,`, `;` o tabuladores,
- ignora encabezados típicos (`source,target`, `src,dst`),
- soporta tanto identificadores enteros como etiquetas de texto, mapeadas a IDs internos,
- elimina lazos ($u = v$) y aristas duplicadas mediante una tabla hash.

Si el archivo no existe o no se pueden extraer aristas válidas, el sistema genera un grafo sintético: un anillo con algunas aristas de largo alcance, útil como caso de prueba reproducible.

4.2. Coloración por comunidades (k-means++)

Para resaltar comunidades visuales se aplica k-means++ sobre las posiciones actuales, usando las posiciones suavizadas cuando el filtro EMA está activo. El número de comunidades K es ajustable y se colorean los nodos con una paleta discreta de doce colores.

Este mecanismo no pretende reemplazar algoritmos de clustering más sofisticados; su objetivo es proporcionar una coloración rápida, reproducible y suficientemente estable para presentaciones interactivas.

4.3. HUD y visión general del sistema

El *HUD* en 2D, dibujado sobre el framebuffer con coordenadas de pantalla, muestra:

- número de nodos N y aristas $|E|$,
- grado máximo,
- FPS estimados en una ventana de tiempo móvil,
- desplazamiento promedio entre iteraciones (Δ),
- temperatura T, estado del *cooling* y número de iteraciones por cuadro,
- modo de fuerzas activo (FR, LinLog, SE) y parámetros de SE cuando aplica,
- estado del suavizado (EMA) y del sistema de iluminación.

La Fig. 1 muestra una vista general del layout en 3D con HUD y coloración por comunidades sobre el grafo sintético.

Un panel adicional del HUD muestra un resumen de los controles de teclado más importantes (cambio de modo, temperatura, zoom, rotación, captura de pantalla, etc.), lo que facilita el uso en demostraciones sin necesidad de documentación externa.

5. Implementación y controles interactivos

La aplicación se implementó en C++ utilizando OpenGL clásico y GLUT para la gestión de ventanas y eventos. El layout se actualiza en la función de *idle* de GLUT, mientras que el dibujo se realiza en la función `display`, aplicando:

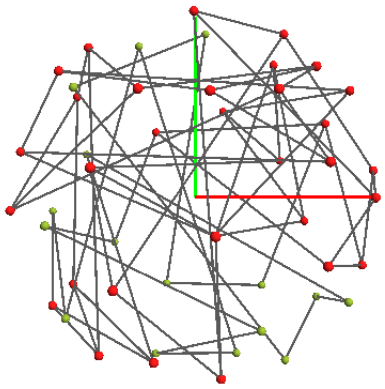


Figura 1: Vista general del sistema: grafo sintético en 3D con coloración por comunidades, HUD con métricas e iluminación activada.

- una cámara fija mirando al origen desde el eje $+Z$,
- rotaciones del *modelo* alrededor de los ejes X e Y,
- iluminación difusa y especular sencilla con una única fuente direccional,
- nodos renderizados como esferas sólidas con radio dependiente del grado.

Los controles de teclado incluyen, entre otros:

- **Espacio**: pausar/reanudar la simulación.
- **t/T**: subir/bajar la temperatura.
- **g**: activar/desactivar *cooling*.
- **m, 1, 2, 3**: cambiar entre FR, LinLog y SE.
- **u/U, y/Y, h/H**: ajustar C_r, C_a y L en el modo SE.
- **c**: alternar coloración por grado / comunidades.
- **s**: activar/desactivar suavizado EMA.
- **F**: ejecutar un preset de demo reproducible (reset, k-means y algunas aristas "puente").
- **Flechas**: rotar el modelo; `[` y `]`: zoom.

El sistema también ofrece una captura de pantalla sencilla en formato PPM, lo que permite generar figuras reproducibles para el informe.

6. Resultados

Se evaluó el comportamiento cualitativo de los tres modelos de fuerzas en dos escenarios:

1. Un grafo sintético de $N = 48$ nodos con estructura de anillo y algunas aristas de largo alcance.
2. El grafo del Karate Club de Zachary con sus dos facciones principales.

6.1. Grafo sintético

En el grafo sintético, FR tiende a producir un anillo relativamente compacto, donde las aristas de largo alcance se manifiestan como "atajos" que cruzan el interior de la estructura. LinLog, en cambio, separa ligeramente los nodos que concentran más aristas de largo alcance, generando bloques algo más definidos.

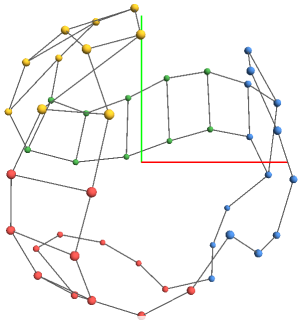
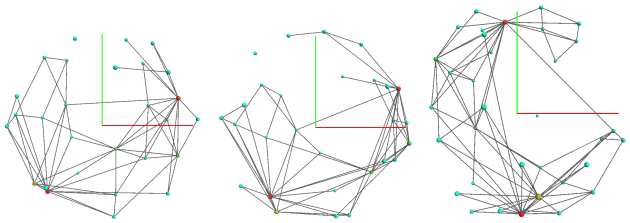


Figura 2: Coloración por comunidades mediante k-means++ sobre el layout en 3D. Cada color representa un clúster geométrico.



(a) FR (b) LinLog (c) SE

Figura 3: Comparación de layouts en 3D para el Karate Club de Zachary con los tres modelos de fuerzas implementados.

En el modo SE, al incrementar C_r y la longitud ideal L se obtiene una configuración donde las aristas de largo alcance estiran el grafo y los nodos periféricos se separan más del núcleo. Esta flexibilidad de parámetros permite ilustrar cómo pequeñas modificaciones en el modelo físico cambian la narrativa visual.

La Fig. 2 muestra un ejemplo de coloración por comunidades sobre el layout sintético, donde k-means++ separa bloques bien conectados en la geometría emergente.

6.2. Karate Club de Zachary

En el caso del Karate Club, los tres modelos revelan las dos facciones clásicas, pero con matices:

- **FR**: genera un layout compacto donde las dos facciones forman agregados cercanos, con un "valle" intermedio donde aparecen algunos nodos puente.
- **LinLog**: refuerza la separación entre las facciones, aumentando la distancia relativa entre los dos bloques principales y haciendo más visibles los brokers que conectan ambos lados.
- **SE**: permite ajustar manualmente el espaciado. Valores altos de C_r y L tienden a abrir un "canal" claro entre las facciones, lo que resulta útil para resaltar los nodos de frontera.

La Fig. 3 resume visualmente estas diferencias para el mismo grafo y parámetros de simulación comparables.

En términos de rendimiento, el sistema mantiene tasas de refresco interactivas para estos tamaños de grafo, con FPS suficientes para rotar y ajustar parámetros en vivo sin pérdida de fluidez perceptible.

7. Discusión

Los resultados sugieren que no existe un modelo de fuerzas universalmente superior; cada uno ofrece ventajas dependiendo de la tarea:

- FR es un *baseline* robusto, con pocos parámetros y layouts previsibles.
- LinLog es preferible cuando el énfasis está en la separación clara de comunidades y en la identificación de puentes.
- SE es especialmente útil cuando se desea controlar el espaciado de manera directa, por ejemplo, para preparar una figura para una presentación o paper.

El paso a 3D permite separar mejor módulos, pero introduce nuevos retos: la oclusión y la necesidad de rotar constantemente la escena para entender la estructura. La cámara fija combinada con la rotación del modelo y el auto-*rotate* mitigaron en gran medida este problema.

Desde el punto de vista pedagógico, el sistema cumple un doble rol: por un lado, sirve como ejemplo compacto de integración de modelos físicos con OpenGL; por otro, permite experimentar de forma interactiva con parámetros que usualmente se presentan sólo de manera teórica en cursos de grafos y visualización.

8. Conclusión y trabajo futuro

Este proyecto presentó un sistema de layout de grafos en 3D basado en modelos de fuerzas intercambiables, implementado en C++ con OpenGL y GLUT. El sistema integra:

- tres modelos de fuerzas (FR, LinLog y Spring–Electrical),
- un módulo de carga de grafos desde CSV genérico,
- un HUD con métricas y controles visibles,
- coloración por comunidades mediante k-means++,
- y una interfaz interactiva que permite explorar parámetros físicos en tiempo real.

Como trabajo futuro se plantean varias extensiones naturales:

- incorporar técnicas de aceleración para la repulsión (p. ej. Barnes–Hut o grids en 3D) para escalar a grafos más grandes;
- añadir soporte para grafos dinámicos (aparición/desaparición de nodos y aristas en tiempo real);
- integrar métricas más sistemáticas de calidad de layout (distorsión de distancias, estrés, alineación con comunidades conocidas);
- explorar representaciones híbridas 2D/3D y proyecciones automáticas para vistas "planas" más legibles.

En conjunto, el sistema demuestra que es posible im-

plementar un pipeline razonablemente completo de layout de grafos en 3D, con control físico e interactividad, en un código suficientemente compacto como para ser analizado y modificado en un contexto de curso avanzado de visualización.

Referencias

[1] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," *Software: Practice and Experience*, 21(11), 1991.

[2] A. Noack, "Energy Models for Graph Clustering," *Journal of Graph Algorithms and Applications*, 11(2), 2007.

[3] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, 31(1), 1989.

[4] P. Gajer, M. Goodrich and S. Kobourov, "A Multi-dimensional Approach to Force-directed Layouts," *Graph Drawing*, LNCS 1984, 2000.

[5] W. W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of Anthropological Research*, 33(4), 1977.