

Dibujado de grafos mediante Stress Majorization

Oscar Esaú Peralta Rosales

12/17/2019

Proyecto final - Métodos Numéricos

El principal objetivo en el campo de Visualización de Datos es ayudar a descubrir, interpretar y comparar relaciones e información existente en un conjunto de datos a través de distintos medios visuales, y uno de los medios que se analizan en este proyecto son los grafos.

Gran variedad de características presentes en los datos pueden ser modeladas a través de grafos, y gracias a su enorme versatilidad, diversos campos de estudio hacen de su uso; geografía, ciencias sociales, movilidad, telefonía, bases de datos, inteligencia artificial, medicina, electrónica, química, etc.

Dadas las características modeladas de los datos a través de los grafos, podemos decir que cada grafo es único y por tanto la forma de como dibujarlo de tal manera que se preserve la estructura de sus relaciones también es variada. Diversas técnicas se han realizado con este fin como, *Stochastic Gradient Descent*, *gradient projection*, *Weighted Constraint Relaxation*, *Multidimensional Scaling* y el método implementado aquí *Stress Majorization*.

Así, a través del proceso de *Stress Majorization* se buscó una forma de poder dibujar este modelo tal que podamos identificar más fácilmente las relaciones en él conservando la estructura intrínseca del grafo y las similitudes existentes entre los nodos.

Introducción

Un grafo $G = \{V, E\}$ se define como un conjunto de vértices V y aristas E en donde cada nodo mapea alguna entidad o característica de los datos y las aristas las relaciones entre estos. Durante este proyecto nos concentramos a trabajar sobre grafos no dirigidos, es decir, las relación presente por una arista es bidireccional.

Stress Majorization es una técnica de optimización usada comúnmente para desplegar información almacenada en una matriz de distancias, la cual contiene la distancia entre cada par de nodos. Esta técnica trata de colocar cada nodo dentro de un *layout* multidimensional tal que la distancias sean preservadas de la mejor manera a través de la función de coste *stress*. La función de coste se define como:

$$stress(X) = \sum_{i < j} w_{ij} (||X_i - X_j|| - d_{ij})^2 \quad (1)$$

Dónde X es un *layout* X de dimensiones $N \times R$ con N como el número de nodos y R la dimensionalidad de este. d_{ij} es la distancia entre el nodo i y el nodo j y w_{ij} es una constante de normalización igual a $d_{ij}^{-\alpha}$.

Así, dado un *layout* inicial X se busca minimizar la función de *stress* para encontrar un nuevo *layout* tal que se tenga un dibujado óptimo.

Stress Majorization

Dado un *layout* X de dimensiones $N \times R$ con $R = 2$ o $R = 3$ se busca minimizar la función de *stress*

$$stress(X) = \sum_{i < j} w_{ij} (||X_i - X_j|| - d_{ij})^2$$

Expandiendo la función de *stress* se tiene

$$stress(X) = \sum_{i < j} w_{ij} d_{ij}^2 + \sum_{i < j} w_{ij} \|X_i - X_j\|^2 - 2 \sum_{i < j} \delta_{ij} \|X_i - X_j\| \quad (2)$$

Con $\delta_{ij} = w_{ij} d_{ij}$ y $w_{ij} = d_{ij}^{-\alpha}$ con $\alpha = 2$ (proporciona mejores resultados).

El segundo término de (2) se puede escribir como

$$\sum_{i < j} w_{ij} \|X_i - X_j\|^2 = (X^T L^w X)^T \quad (3)$$

Donde L^w es la forma cuadrática de la Matriz Laplaciana Ponderada de dimensiones $N \times N$

$$L_{ij}^w = \begin{cases} -w_{ij} & i \neq j \\ \sum_{k \neq i} w_{ik} & i = k \end{cases} \quad (4)$$

El tercer término puede ser limitado, usando la desigualdad de *Cauchy-Schwartz*

$$\|x\| \|y\| \geq x^T y$$

y adicionalmente dada una matriz Z de las mismas dimensiones de X tenemos

$$\|X_i - X_j\| \|Z_i - Z_j\| \geq (X_i - X_j)^T (Z_i - Z_j) \quad (5)$$

$$\sum_{i < j} \delta_{ij} \|X_i - X_j\| \geq \sum_{i < j} \delta_{ij} (\|Z_i - Z_j\|)^{-1} (X_i - X_j)^T (Z_i - Z_j) \quad (6)$$

La ecuación (6) puede ser reescrita de forma matricial como

$$\sum_{i < j} \delta_{ij} \|X_i - X_j\| \geq (X^T L^Z Z)^T \quad (7)$$

Dónde dada una matriz Z de posiciones de las mismas dimensiones de X podemos construir L^Z como

$$L_{ij}^Z = \begin{cases} -\delta_{ij} (\|Z_i - Z_j\|)^{-1} & i \neq j \\ \sum_{k \neq i} L_{ik}^Z & i = k \end{cases} \quad (8)$$

De esta forma la función *stress* puede ser limitada usando las ecuaciones anteriores

$$stress(X) \leq \sum_{i < j} w_{ij} d_{ij} + (X^T L^w X)^T + 2(X^T L^Z Z)^T \quad (9)$$

Diferenciando con respecto a X se obtiene que el mínimo de la función de stress (9) está dado por

$$L^w X = L^Z Z \quad (10)$$

equivalentemente resolviendo para cada eje de X

$$L^w X^{(a)} = L^Z Z^{(a)} \quad a = 1, 2, 3 \dots R \quad (11)$$

La ecuación (11) puede formularse de manera iterativa para un *layout* inicial $X(t)$ y obtener un nuevo *layout* $X(t+1)$ tal que satisfaga $stress(X(t+1)) \leq stress(X(t))$

$$L^w X(t+1)^{(a)} = L^{X(t)} X(t)^{(a)} \quad a = 1, 2, 3 \dots R \quad (12)$$

El proceso iterativo se detiene mediante la siguiente condición

$$\frac{stress(X(t)) - stress(X(t+1))}{stress(X(t))} \leq \epsilon \quad (13)$$

comúnmente con $\epsilon \sim 10e^{-4}$. Nótese que L^w es constante durante todo el proceso y $L^{X(t)}$ depende del *layout* actual y que las matrices formadas en el proceso derivadas del Laplaciado son semidefinidas positivas.

Implementación y Resultados

Conjuntos de Datos

Los conjuntos de datos fueron obtenidos desde la página de The SuiteSparse Matrix Collection elegidos por cantidad de nodos y distribución de los nodos.

Table 1: Datasets usados para dibujado

Nombre	Nodos	Tipo
1138_bus	1138	Power Network Problem
cage9	3534	Directed Weighted Graph
CSphd	1882	Directed Graph
dwt_1005	1005	Structural Problem
g_26	2000	Undirected Random Graph
gre_216a	216	Directed Weighted Graph
gre_1107	1107	Directed Weighted Graph
qh882	882	Power Network Problem

Generación del layout inicial

El *layout* solo cuenta con dos dimensiones por tanto la matriz X es de dimensión $N \times 2$. La primer columna de la matriz representa el eje x y la segunda columna el eje y. Así el i -ésimo nodo está posicionado en las coordenadas $(X[i][0], X[i][1])$.

Los valores de las coordenadas iniciales para cada nodo son tomados aleatoriamente y posteriormente cada columna es normalizada tener en la menor manera posible errores numéricos.

Cálculo de las distancias entre los nodos

En la literatura sobre diversas implementaciones se observa que el cálculo de la distancia de los nodos lo realizan a través del *Algoritmo de Dijkstra*, el cual nos ayuda a obtener la distancia de un nodo hacia los demás. Sin embargo, necesitamos la distancia de todos los nodos hacia todos los nodos y así poder construir una matriz de distancias D , dónde la intersección de la fila i con la columna j representa la distancia que hay entre los dos nodos i y j se procedió a implementar el algoritmo de *Floyd Warshall*, el cuál hace uso de de la aproximación *Bottom-Up* de programación dinámica para realizar el cálculo de todas estas distancias de una sola vez.

Implementación del algoritmo

Como se vio anteriormente, la solución de la función *stress* está dada por

$$L^w X(t+1)^{(a)} = L^{X(t)} X(t)^{(a)} \quad a = 1, 2, 3 \dots R$$

Con las matrices L tal como se definen arriba. A continuación se muestra el pseudocódigo de la implementación.

```

1 function graph_layout_solver1(n_nodes, graph, niters, tolerance) {
2   // Inicializar posiciones de los nodos aleatoriamente
3   X <- graph_build_layout_matrix(n_nodes);
4
5   // Calcular matriz de distancias mediante Floyd Warshall
6   distances <- get_distances(graph, n_nodes);
7
8   // Construir matriz delta delta_ij = w_ij * d_ij
9   delta_matrix <- get_delta_matrix(distances);
10
11  // Construir matriz laplaciana ponderada con las distancias
12  Lw <- get_laplacian_w_matrix(distances);
13  A <- Lw
14  for (iter = 0; iter < niters; iter++) {
15    Z <- X;
16    // Obtenemos la matriz laplaciana de Z
17    Lz <- _get_laplacian_z_matrix(Z, delta_matrix);
18
19    for (axe = 0; axe < 2; axe++) {
20      // Calculamos b como el producto matricial de Lz por el eje actual de X
21      b = Lz @ X[axe]
22      // Resolver Ax=b
23      X[axe] <- solve(A, b);
24    }
25    // Criterio de parada
26    stress_tol = (stress(distances, Z) - stress(distances, X)) / stress(distances, Z);
27    if (abs(stress_tol) < tolerance) {
28      break;
29    }
30  }
31
32  return X;

```

Así, la solución está dada por resolver iterativamente un sistema lineal $Ax = b$ generado consecutivamente por las matrices Laplacianas y el *layout* actual obtenido. La solución de estos sistemas fue encontrada con el uso de 5 métodos de solución; Método de Cholesky, Método de Doolittle, Método de Gauss-Seidel, Método de Jacobi, y el Método de Gradiente Conjugado.

A continuación se presentan una comparativa de entre los métodos de solución para el grafo formado a partir del conjunto de datos *dwt_1005*.

Table 2: Resultados de los métodos internos de solución para el dataset *dwt_1005*

Método	Iteraciones	Tiempo de ejecución	Criterio de convergencia alcanzado
Cholesky	13	3.117s	5.0e-06
Doolittle	25	3.665s	4.4e-06
Gauss-Seidel	13	90.189s	4.9e-06
Jacobi	15	109.634s	8.8e-06
Conjugate Gradiente	4	25.870s	4.8e-06

Como se observa en la tabla de resultados (2) el método que mejor se comporta fue el de Cholesky con un tiempo promedio de 3.117 segundos y 13 iteraciones necesarias para la convergencia. El método de Doolittle a pesar de necesitar mayor número de iteraciones de convergencia su tiempo promedio de solución es muy similar para Cholesky, ambos con resultados muy similares en el dibujo del grafo.

Los siguientes dos métodos iterativos, Gauss-Seidel y Jacobi también obtuvieron muy buenos resultados, sin embargo su tiempo de convergencia incrementó drásticamente. A pesar de que el proceso iterativo de la minimización de la función *stress* llegó a converger no se logró una convergencia en las soluciones parciales de los sistemas de ecuaciones generados, generando *layouts* nada prometedores.

Las siguientes figuras muestran una comparativa entre el layout obtenido con los métodos descritos anteriormente contra la herramienta de dibujo de grafos graph-tool.

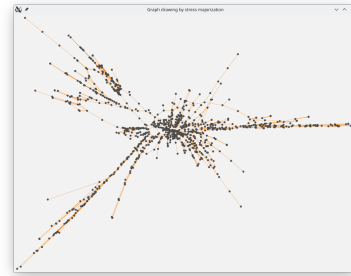
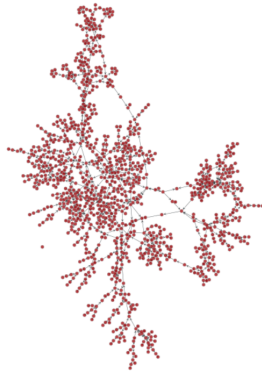


Figure 1: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset 1138__bus.

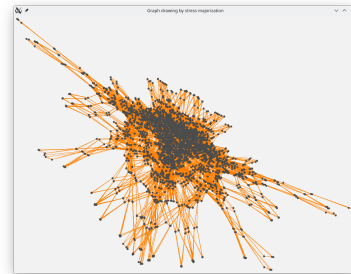
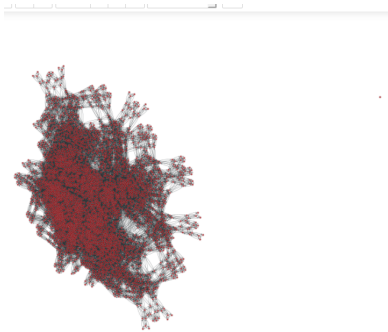


Figure 2: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset cage9.

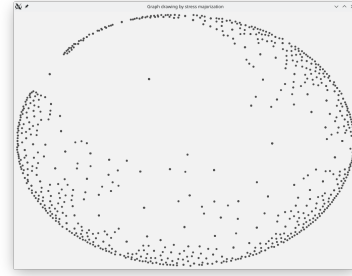
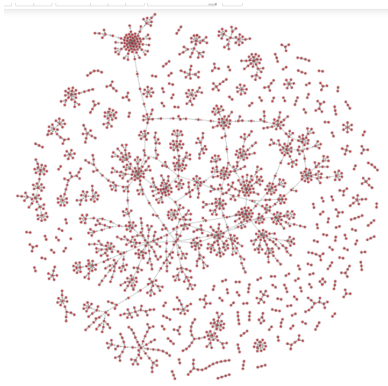


Figure 3: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset CSphd.

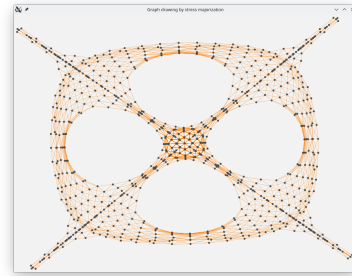
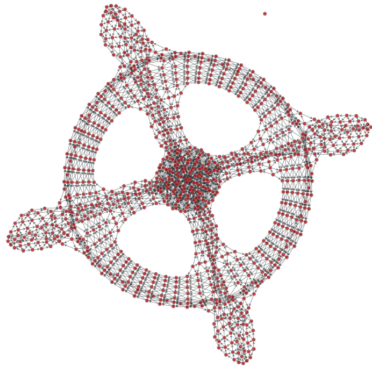


Figure 4: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset dwt_1005.

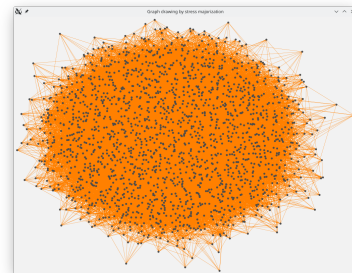


Figure 5: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset G26.

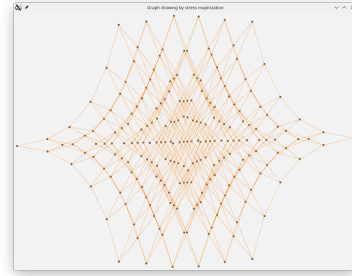
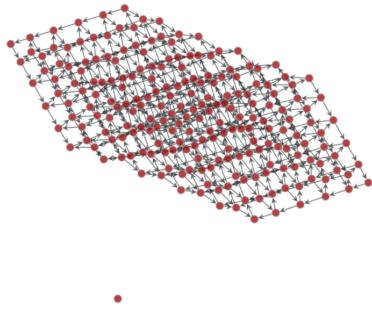


Figure 6: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset gre_216a.

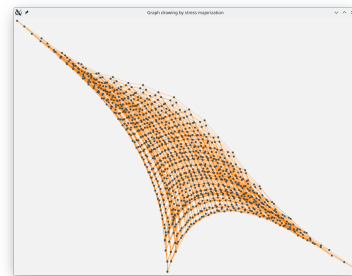
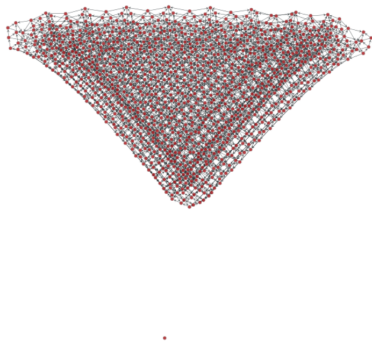


Figure 7: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset gre_1107.

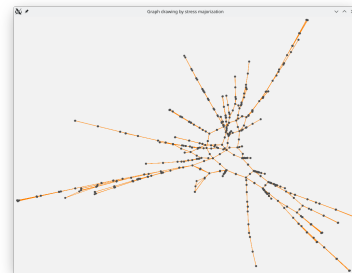
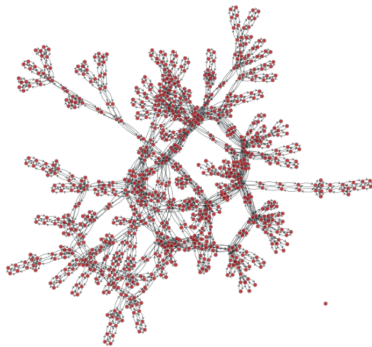


Figure 8: A la izquierda layout obtenido usando Graph-tool a la derecha el layout obtenido en la implementación del dataset qh882.

Conclusiones

Dado que las matrices usadas durante el proceso de solución derivan de la matriz Laplaciana y como se menciona en [1] son semidefinidas positivas. Así que los mejores métodos de solución propuestos son el Método de Cholesky y el Método de Gradiente Conjugado.

Al igual que se menciona en diversos artículos se corroboró que el mejor método de solución para la minimización de la función *stress* fue el método de Cholesky. Como se vio en los resultados el tiempo de convergencia y número de iteraciones fue superior (en términos de rapidez) a los demás algoritmos probados.

Sin embargo, no se obtuvieron los resultados deseados con el Método de Gradiente a pesar de que los artículos mencionan su uso. Durante su ejecución se observaba una convergencia entre el criterio de parada dado por el cálculo de la función *stress* con el *layout* anterior y el *layout* actual, pero a pesar de los esfuerzos la convergencia de la solución de los sistemas de la forma $Ax = b$ generados internamente no ocurrió.

Los resultados obtenidos y mostrados en las distintas figuras comparativas son bastante buenos, aunque aún lejos de los resultados posibles usando herramientas profesionales para este tipo de problemas.

Referencias

- Emden R. Gansner, Yehuda Koren and Stephen North, Graph Drawing by Stress Majorization, Springer 2004
- Tim Dwyer, Yehuda Koren, Kim Marriott, Constrained graph layout by stress majorization and gradient projection, 13th International Symposium on Graph Drawing, 2005