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 estos 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 artififial, medicina, electrónica, química, etc. Así, a través del proceso de *Stress Majorization* se buscó la 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 un conjunto de vértices V y aristas E. en dónde cada nodo mapea alguna entidad o característica de los datos y las aristas las relaciones entre estos. Durante este trabajo nos concetramos a trabajar sobre grafos no dirigidos, es decir, las relación presente por una arista es bidireccional.

Dadas las características modeladas de los datos a través de los grafos, podemos intuir 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 proyection*, *Weighted Constraint Relaxation*, *Multidimensional Scaling* y el método implementado aquí *Stress Majorization.*

Stress Majorization es una técnica de optimización usada comúnte para desplegar información almacenada en una matrix de distancias, la cual contiene la distancia entre cada par de nodos. Esta técnica trata de colocar cada nodo dentro de un *layout* multidimencional 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:

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

Dónde X es un *layout* X de dimenciones NxR 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 dimenciones NxR 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

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

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

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

Donde L^w es la forma cuadrática de la Matriz Laplaciana Ponderada de dimensiones NxN

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

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 dimenciones de X tenemos

$$||X_i - X_j|| ||Z_i - Z_j|| \geq (X_i - X_j)^T (Z_i - Z_j) \tag{5}$$

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

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

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

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

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

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

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

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

$$L^w X = L^Z Z \tag{10}$$

equivalentemente resolviendo para cada eje de X

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

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 $stress(X(t+1)) \leq stress(X(t))$

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

El proceso iterativo se detiene mediante la siguiente condición

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

con $\epsilon \sim 10e^{-}4$. Nótese que L^w es constante durante todo el proceso y $L^{X(t)}$ depende del layout actual.

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
dwt_2680	2680	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 $Nx2$. 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 aleatoreamente 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 *Algorimo 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 algortimo de *Floyd Warshall.* el cuál hace uso de de la aproximación *Buttom-Up* de programación dinámica para realizar el cálculo de todas estas distancias de unas sola vez.

Implementación del algoritmo

Como se vió anteriomente la solución de la función *stress* está dada por

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

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

```
function graph_layout_solver1(n_nodes, graph, niters, tolerance) {
    // Inicializar posiciones de los nodos aleatoreamente
    X <- graph_build_layout_matrix(n_nodes);

    // Calcular matriz de distancias mediante Floyd Warshall
    distances <- get_distances(graph, n_nodes);

    // Construir matriz delta delta_ij = w_ij * d_ij
    delta_matrix <- get_delta_matrix(distances);

    // Construir matriz laplaciana ponderada con las distancias
    Lw <- get_laplacian_w_matrix(distances);
    A <- Lw
    for (iter = 0; iter < niters; iter++) {
        Z <- X;
        // Obtenemos la matriz laplaciana de Z
        Lz <- _get_laplacian_z_matrix(Z, delta_matrix);

        for (axe = 0; axe < 2; axe++) {
            // Calculamos b como el producto matricial de Lz por el eje actual de X
            b = Lz @ X[axe]
            // Resolver Ax=b
            X[axe] <- solve(A, b);
        }
        // Criterio de parada
        stress_tol = (stress(distances, Z) - stress(distances, X)) / stress(distances, Z);
        if (abs(ctol) < tolerance) {
            break;
        }
    }

    return X;
```

Así, la solución está dada por resolver iterativamente un sistema lineal $Ax = b$ generadas por las matrices Laplacianas y el *layout* generado. La solución de estos sistemas fué 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.

Table 2: Resultados de los métodos internos de solución

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