Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Algoritmo y heurística para incrustar métricas en una línea

Proyecto de investigación
Tercera versión

Trimestre 2016 Invierno

Saúl Martínez Juárez
2113000992
Asesores de proyecto terminal

Dr. Francisco Javier Zaragoza Martínez
Profesor Titular
Departamento de Sistemas

4 de mayo de 2016

En caso de que el Comité de Estudios de la Licenciatura en Ingeniería en Computación apruebe la realización de la presente propuesta, otorgamos nuestra autorización para su publicación en la página de la División de Ciencias Básicas e Ingeniería.

Saúl Martínez Juárez

Asesor
Dr. Francisco Javier Zaragoza Martínez

---

1. Introducción

Una función $d : X \times X \to \mathbb{R}$ es una métrica en un conjunto no vacío X si se satisfacen las siguientes condiciones para todo $x, y, z \in X$:

- Positividad: $d(x, y) > 0$; si $x \neq y$, y $d(x, x) = 0$.
- Simetría: $dc = d(y, x)$.
- Desigualdad del triángulo: $d(x, z) \leq d(x, y) + d(y, z)$.

A $d(x, y)$ se le llama la distancia de x a y. Por poner dos ejemplos de métricas, si $X = \mathbb{R}^n$ y $x = (x_1, x_2, ..., x_n)$, $y = (y_1, y_2, \ldots, y_n)$, entonces la métrica euclideana se calcula como:

$$d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

y la métrica Manhattan se calcula como:

$$d_M(x, y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|.$$

Sea d_a una métrica en A y d_b una métrica en B. Entonces una función $f : A \to B$ es una incrustación con contracción c_f y expansión e_f si para cada par de puntos $p, q \in A$ se cumple que

$$\frac{d_a(p, q)}{c_f} \leq d_b(f(p), f(q)) \leq e_f \cdot d_a(p, q)$$

.

Adicionalmente diremos que f no contrae si $c_f \leq 1$. Una función f que no contrae tiene distorsión α si $e_f \leq \alpha$.

La compresión de datos es la codificación de un cuerpo de datos D en un cuerpo de datos más pequeño D'. Una parte central en la compresión es la redundancia en los datos. Sólo los datos con redundancia pueden comprimirse aplicando un método o algoritmo de compresión que elimine o remueva de alguna forma dicha redundancia. La redundancia depende del tipo de datos (texto, imágenes, sonido, etc), por tanto, no existe un método de compresión universal que pueda ser óptimo para todos los tipos de datos. La compresión puede ser sin pérdida (reversible) o con pérdida (irreversible).
La expansión de una métrica es equivalente a su distorsión, y también equivalente a una compresión con pérdida. La razón por la que vale la pena tener una distorsión, es reducir la complejidad de un espacio. Dependiendo de su uso posterior, algunos espacios son más complejos que otros.

Ejemplo: Considere la gráfica de tamaño 3 con pesos en las aristas de la figura 1, que intentamos colocar en una línea recta, en el primer intento las colocamos en orden numérico, y la longitud total es 7 (ver figura 2). Podemos mejorarlo y

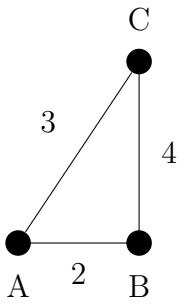

Figura 1: Gráfica original de tamaño 3.

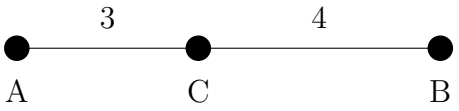

Figura 2: La longitud total de la línea es $z = 7$ ($\alpha = \frac{3+4}{2} = \frac{7}{2} = 3.5$).

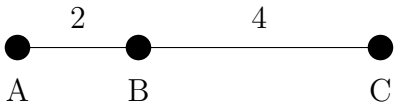

Figura 3: La longitud total de la línea es $z = 6$ ($\alpha = \frac{2+4}{3} = \frac{6}{3} = 2$).

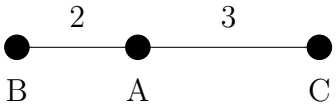

Figura 4: La longitud total de la línea es $z = 5$ ($\alpha = \frac{3+2}{4} = \frac{5}{4} = 1.25$)

en el siguiente intento, la longitud es 6 (ver figura 3). Sin embargo, este resultado aún puede mejorarse al incrustar en una línea de longitud 5 (ver figura 4). Observe que en estos ejemplos, a menor longitud, también menor distorsión α.

Entonces, el problema de incrustamiento de una métrica en una línea con distorsión mínima es un problema de optimización en el que incrustamos una métrica en una línea recta con el menor espacio posible.

En este trabajo se implementará un método exacto y un método heurístico para resolver el problema de incrustación de una métrica en una línea con la menor longitud posible.

2. Antecedentes

Hay algunos problemas que se asemejan o con los que comparte algunas características:

El problema de floor layout [8]: Distribución de suelos es un problema que enfrenta la ingeniería industrial a maximizar la producción por unidad de suelo.

El problema de pooling [7]: La administración de recursos en un fondo común cercano para evitar requerirlos de un lugar más costoso, siendo estos lo más heterogéneos posible.

También se parece al problema del agente viajero [6], con la diferencia de que la gráfica inicial de la que buscaremos el recorrido no está inicial ni estáticamente ponderada.

El proyecto terminal Encajes primitivos de gráficas planares exteriores [1] busca incrustar grafos en espacios (gráficas) bidimensionales. Se parece al proyecto pues logra hacerlo en un espacio reducido, acercándose al mínimo en tiempo polinomial, lo cual es un buen precedente. Se diferencia en que las gráficas son estrictamente planares y en este trabajo son métricas.

El proyecto terminal Encajes primitivos de árboles planos [2] incrusta árboles, que son un subconjunto de las gráficas, se diferencia en que dado que son árboles, no siempre hay un camino directo de un vértice a otro sino que hay una trayectoria de por medio. Esto es un caso especial de nuestro problema en el que las trayectorias son de tamaño 1.

El proyecto terminal Diseño de reglas de Golomb óptimas [3] nos ayuda mucho en el diseño de algoritmos en la fase entera, ya que hay buenos resultados en este campo y especialmente en este proyecto, pues genera una serie que podemos reutilizar. Sin embargo, en nuestro proyecto no siempre se generan reglas de Golomb, cosa que es requisito en esta referencia.

Finalmente el problema del incrustamiento con mínima distorsión [4], es el más parecido pues sólo hay que declarar que es una métrica lo que se va a incrustar.

3. Justificación

El interés principal del problema de incrustación radica en la compresión de datos; los archivos al ser comprimidos pueden ser transportados y almacenados a menor costo. Mientras nuestros recursos sean limitados (no infinitos) nos interesará reducir cualquier tipo de costo. En la actualidad, el trabajo de compresión es muy valorado, y con justa razón; cualquier aplicación de tecnología tiende a utilizar cada vez más información lo cual se convierte en un reto técnico.

Para almacenar métricas lo más común es guardarlas como matrices, simétricas de diagonal 0, y por consiguiente de tamaño $\Theta(n^2)$, donde n representa el total de puntos a representar. Este valor polinomial es el que es interesante reducir: si se incrusta la métrica en un espacio tridimensional, la métrica total termina midiendo $\Theta(n)$, por lo que el tamaño se redujo drásticamente. Lo mismo sucede con espacios bidimensionales y unidimensionales, como es el caso actual. Cualquier incrustación de métricas en espacios euclideanos n-dimensionales reducen la complejidad del espacio necesario para representarlas, por lo que la incrustación es necesariamente un método de compresión eficiente.

En ocasiones, métricas específicas no son las adecuadas para ciertas operaciones, por lo que es necesario traducirlas o interpretarlas hacia otra métrica, la conversión no es completamente biyectiva, por lo que tiende a tener un error, que en lenguaje de métricas se le conoce como distorsión. Para términos prácticos de compresión de los datos, esto es pérdida de información.

El producto final de este proyecto contendrá especificaciones sobre el modo de manejar problemas de optimización en los que el conjunto de restricciones está incluída directamente en el producto final, que en este caso es una matriz

incrustada en una línea recta.

La complejidad computacional de este problema permanece abierta, y los resultados de este proyecto pueden contribuir al estudio de los problemas NP y a su clasificación.

4. Objetivos

4.1. Objetivo general

- Diseñar, implementar y evaluar un algoritmo exacto y una heurística para el problema de incrustación de una métrica en una línea.

4.2. Objetivos específicos

1. Diseñar e implementar los algoritmos que generen métricas.
2. Diseñar e implementar un algoritmo exacto que resuelva el problema.
3. Diseñar e implementar un algoritmo heurístico que resuelva el problema.
4. Evaluar los resultados de ambos algoritmos.

5. Metodología

5.1. Generadores de métricas

A continuación proponemos 3 módulos distintos que generarán una matriz A correspondiente a una métrica.

euclideano(n, d, l): Recibe enteros n, d y l. Se generan n puntos numerados $p_1, p_2 \dots p_n$ que están en un espacio d-dimensional con coordenadas aleatorias enteras entre 1 y l. Devuelve una matriz A de $n \times n$ con entradas A_{ij} iguales a la distancia euclideana entre el punto p_i y el punto p_j.

manhattan(n, d, l): Recibe enteros n, d y l. Se generan n puntos numerados $p_1, p_2 \dots p_n$ que están en un espacio d-dimensional con coordenadas aleatorias enteras entre 1 y l. Devuelve una matriz A de $n \times n$ con entradas A_{ij} iguales a la distancia Manhattan entre el punto p_i y el punto p_j.

$grafica(n, l)$: Recibe dos enteros n y l. Genera la gráfica completa K_n de aristas con costo aleatorio entero entre 1 y l. Devuelve una matriz A de $n \times n$ con entradas A_{ij} iguales al costo mínimo de ir del vértice v_i al vértice v_j.

5.2. Algoritmo exacto

Para hacer un algoritmo exacto podemos generar todas las permutaciones de los puntos sobre la recta y verificar todas las restricciones de no contracción. La

solución exacta será la permutación con menor distancia entre sus extremos. Esto implica que para el peor de los casos la complejidad del algoritmo es de $\Theta(n! \cdot n)$ [4]. Es una cifra peor que exponencial, pero nos garantiza encontrar el óptimo. Se puede distribuir el trabajo en 3 módulos:

$genPerm(n, T)$: Recibe dos enteros n y T. Genera la permutación número T de las $n!$ permutaciones de los enteros del 1 al n en forma de vector s.

$calcSol(s, A)$: Recibe una permutación s y una matriz A. Calcula la longitud mínima necesaria para encajar los n puntos en el orden de la permutación s. Devuelve el valor resultante del cálculo.

$evalua(n, A)$: Recibe un entero n y una matriz A. Se invoca el método $genPerm(n, i)$ para asignar una permutación, luego evalúa el resultado invocando a $calcSol(s, A)$ comparándolo con el mejor hasta el momento. Si el nuevo resultado es mejor, guarda el resultado y el vector. Devuelve la longitud mínima encontrada.

5.3. Algoritmo heurístico

Proponemos algoritmos glotones para resolver el problema. Todos los algoritmos glotones tendrán la misma interfaz, cada uno de ellos escogerá un orden para los puntos.

$gloton(n, A)$: Recibe un entero n, y una matriz A, devuelve una permutación s de tamaño n sobre la mejor solución encontrada por el algoritmo.

La permutación encontrada por el algoritmo será evaluada en una llamada a $calcSol(s, A)$ ya explicada anteriormente.

6. Especificación técnica

Para la implementación de este proyecto se toman en cuenta factores que influyen directamente en el comportamiento y éxito de los módulos individuales:

$euclideano(n, d, l)$: El rango de valores de n es de 3 hasta 1000, el de d es entre 2 y 10 y el rango de l está entre 1 y 100. La distribución aleatoria es uniforme. Las entradas de la matriz A están redondeadas al entero superior.

$manhattan(n, d, l)$: El rango de valores de n es de 3 hasta 1000, el de d es entre 2 y 10 y el rango de l está entre 1 y 100. La distribución aleatoria es uniforme.

$grafica(n, l)$: El rango de valores de n es de 3 hasta 1000 y el rango de l está entre 1 y 100. La distribución aleatoria es uniforme. El camino mínimo de la gráfica se obtiene aplicando el algoritmo de $Floyd - Warshall$

$genPerm(n, T)$: El rango de valores de n es de 3 hasta 15 y el rango de T está entre 1 y 15!.

$calcSol(s, A)$: El rango de valores de los elementos de la permutación s es entre 1 y 15, el tamaño de la s está entre 3 y 15. El tamaño de A está entre 3×3 y 15×15 y los elementos de A están entre 1 y 1000.

$evalua(n, A)$: El rango de valores de n está entre 3 y 15. El tamaño de A está entre 3×3 y 15×15 y los elementos de A están entre 1 y 1000.

gloton(n, A): El rango de valores de n es de 3 hasta 1000. El tamaño de A está entre 3×3 y 1000×1000 y los elementos de A están entre 1 y 1000.

Como lenguaje de programación se usará C++14 (g++ 5.0 o superior para Linux).

7. Cronograma de trabajo

La UEA correspondiente a las actividades será Proyecto de Integración en Ingeniería en Computación I. Este proyecto cubrirá un mínimo de 18 horas semanales durante 11 semanas, cumpliendo con un total de 198 horas, las cuales se realizarán durante el lapso del trimestre académico 16-P. Las actividades se encuentran en la tabla 1.

Tabla 1: Primavera 2016

No.	Actividad	Horas	Entregables
1	Diseñar e implementar los módulos que generan métricas.	36	Módulos generadores de métricas.
2	Diseñar e implementar el algoritmo exacto.	72	Código fuente de algoritmo exacto.
3	Diseñar e implementar un algoritmo heurístico.	72	Código fuente de algoritmo heurístico.
4	Evaluar.	18	Resultados y conclusiones.

8. Recursos

Los recursos utilizados son un IDE y compilador para C++14, Codeblocks disponible desde "http://www.codeblocks.org/downloads"[11]

El asesor se responsabiliza de guiar al alumno y de que todos los recursos anteriormente citados estarán disponibles para el alumno, de modo que el proyecto de integración se pueda concluir en tiempo y forma.

Saúl Martínez Juárez

Asesor
Dr. Francisco Javier Zaragoza Martínez

Referencias

[1] J.A. Pérez Arcos, *"Encajes primitivos de gráficas planares exteriores"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[2] C. D. Alfaro Quintero, *"Encajes primitivos de árboles planos"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[3] J. J. Santana González, *"Diseño de reglas de Golomb óptimas"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.

[4] Fedor V. Fomin, Daniel Lokshtanov and Saket Saurabh, *"An Exact Algorithm for Minimum Distortion Embedding"*, Theoretical Computer Science, vol. 412, no. 29, pp 3530-3536, 2011.

[5] Piotr Indyk, *"Algorithmic Applications of Low-distortion Geometric Embeddings"*, FOCS IEEE, p. 40, 2001.

[6] Ríos, R. González, J. *"Investigación de operaciones en acción: Heurísticas para la solución del Problema del Agente Viajero"*. Ingenierías, vol. 3, p. 9, 2000.

[7] Audet, Charles, *"Pooling problem: Alternate formulations and solution methods"*. Management science, vol. 50, no 6, p. 761-776, 2004

[8] Huchete, Joey; Dey, Santanu S.; Vielma, Juan Pablo. *"Beating the SDP bound for the floor layout problem"*, arXiv preprint arXiv:1602.07802, 2016.

[9] D. Salomon, *"David. A guide to data compression methods"*, Springer Science & Business Media, 2013.

[10] Fowler, E. James, R. Yagel, *"Lossless compression of volume data"*, ACM, p. 43-50.