Una Metaheurística GRASP para Integración en Grafos

Manuel Dubinsky
Ingeniería Informática
Dpto. Tecnología y Administración
Universidad Nacional de Avellaneda
Argentina

César Massri
Dpto. de Matemática
Universidad de CAECE,
IMAS, CONICET
Argentina

Fernando Asteasuain
Ingeniería Informática
Dpto. Tecnología y Administración
Universidad Nacional de Avellaneda
Argentina

***Abstract*—Given an edge-weighted graph, we analyze the problem of finding an orientation of its edges and a function on its nodes, such that for each oriented edge the consistent subtraction of the function on its incident vertices (ie.: head - tail), is the best approximation in a least square sense to the original edge-weighted function. We present a simple GRASP algorithm to find a "good" solution that is suitable for distributed execution.**

1. Introducción

1.1. Descripción del problema

Dado un grafo conexo $G = (V, E)$ [Harari, 1969] y una función de pesos en los ejes $f : E \to \mathbb{R}$, queremos hallar una orientación de los ejes de G y una función de pesos en los nodos $x : V \to \mathbb{R}$ de modo de minimizar el error:

$$E(x) := \sum_{e \in E} (dx(e) - f(e))^2$$

donde $dx : E \to \mathbb{R}$ es el *diferencial* de x (asociado a la orientación de los ejes) y se define en cada eje $e = v \to w$ como $dx(e) = x(w) - x(v)$.

Concretamente, si consideramos la correspondencia entre orientaciones de G y *matrices de incidencia dirigida*, se trata de encontrar: por un lado la "mejor" *matriz de incidencia dirigida* D, y por el otro la mejor función de pesos en los ejes $x : V \to \mathbb{R}$ de modo de minimizar la expresión:

$$E(x) = \|D\boldsymbol{x} - \boldsymbol{f}\|^2$$

donde:

$$\boldsymbol{x} = \begin{bmatrix} x(v_1) \\ \dots \\ x(v_n) \end{bmatrix}, \boldsymbol{f} = \begin{bmatrix} f(e_1) \\ \dots \\ f(e_m) \end{bmatrix}$$

El algoritmo que proponemos explora el espacio de todas las orientaciones posibles del grafo original. Nuestra principal contribución son los criterios empleados en la metaheurística, que constituye el diseño de una "norma" (α-norma) para comparar dichas orientaciones.

1.2. Contexto del problema y trabajos relacionados

El problema surgió a partir de la definición de *1-forma diferencial* en el contexto de *mallas poligonales*.

Las formas diferenciales [Spivak, 1965], [Tu, 2008] conceptualizan la noción de objeto integrable. Son objetos de estudio en geometría diferencial. Intuitivamente se puede pensar en una *1-forma diferencial* como una cantidad que mide una distancia infinitesimal y que puede ser integrada a lo largo de una curva. Análogamente una *2-forma diferencial* es una cantidad que mide una superficie infinitesimal y que puede ser integrada a lo largo de una superficie. Y en general, una *k-forma diferencial* está asociada a una variedad diferencial k-dimensional y puede ser integrada en dimensión k. Esta noción permite unificar varios teoremas clásicos como el *teorema de Green* y el *teorema de Stokes*. Además de la naturaleza geométrica de este punto de vista, las formas diferenciales tienen una rica estructura algebraica que permite entre otras cosas trasladar propiedades invariantes entre dos variedades mediante mapas diferenciales. El precursor de la teoría fue el matemático francés *Elie Cartan* a fines del siglo XIX.

Por otro lado, las *mallas poligonales* [Botsch, Kobbelt, Pauly, Alliez, Lévy, 2010] permiten representar superficies inmersas en el espacio tridimensional. Están compuestas por aproximaciones lineales poligonales de pequeñas vecindades de la superficie denominadas caras. Y tienen asociado un grafo cuyos nodos y ejes definen los vértices y lados de las caras. Son objetos de estudio fundamentales en el área de *procesamiento digital de geometría*. Esta área ha crecido de manera sostenida en los últimos veinte años debido a la capacidad de obtener modelos geométricos de objetos reales mediante técnicas como: tomografía computarizada, resonancia magnética, escaneo mediante láser y ultrasonido, y microscopía. Numerosas disciplinas como la medicina, la ingeniería, la astrofísica, la exploración geográfica, la arquitectura y el urbanismo, como así también las industrias cinematográfica, de videojuegos, y de fabricación de manufacturas se nutren continuamente de modelos tridimensionales. El área de *procesamiento digital de geometría* es la encargada de proveer herramientas y aplicaciones para producir dichos modelos. Las aplicaciones son muy variadas, algunos ejemplos son: escaneo 3D [Lanman, Raskar, Agrawal,

Taubin, 2008], navegación de vehículos autónomos [Crispell, Mundy, Taubin, 2008], reconocimiento de patrones [Taubin, Cooper, 1992], reconocimiento de trayectorias [Zhao, Taubin, 2009], construcción de modelos tridimensionales de sitios arqueológicos [REVEAL], de esculturas y objetos antiguos [Taubin et al., 2002], cinematografía 3D [Ronfard, Taubin, 2007], etc.

La teoría de formas diferenciales tiene su contraparte discreta en el marco de las mallas poligonales. Esto se debe a que una malla poligonal puede ser entendida como una descomposición simplicial [Hatcher, 2004] de la superficie que está modelando. Y una forma diferencial puede interpretarse como un complejo de co-cadena sobre dicha descomposición. En el trabajo [Desbrun, Kanso, Tong, 2006], los autores muestran cómo integrar formas diferenciales mediante una técnica de interpolación. En ese contexto los 1-símplices de la descomposición, o dicho de otro modo: los ejes del grafo asociado a la malla, ya están dotados de una orientación. Y están sujetos a la equivalencia $v_0v_1 = -v_1v_0$, es decir que si se considera la orientación contraria de un eje, su término asociado en el complejo de co-cadena cambia de signo (esto es compatible con el primer teorema fundamental del cálculo).

El problema que consideramos en este trabajo es una variante en la cual los ejes del grafo subyacente no están dotados de una orientación. El problema de optimización consiste justamente en encontrar una orientación que ajuste del mejor modo posible los valores de los vértices de acuerdo a una función de pesos en los ejes del grafo. Creemos que la formulación planteada en este trabajo es novedosa. No encontramos antecedentes en la bibliografía y es por esto que no pudimos referirnos a instancias y algoritmos con las cuales compararnos en la fase experimental.

1.3. Aplicaciones

Las aplicaciones son aquellas en las cuales necesitamos construir una función de los nodos que respete ciertas restricciones diferenciales.

Por ejemplo, supongamos que los nodos representan regiones geográficas y queremos construir un mapa topográfico basado en ciertas relaciones diferenciales de altura entre algunas regiones.

Otro ejemplo es el de ordenar cronológicamente ciertos eventos en base a algunas relaciones diferenciales entre ellos.

2. Preliminares

2.1. Álgebra lineal

La *matriz de incidencia dirigida*

La *matriz de incidencia dirigida* (D) de un grafo dirigido $\vec{G} = (\vec{V}, \vec{E})$ es una matriz de $m \times n$ (donde $|\vec{E}| = m$ y $|\vec{V}| = n$) tal que para cada eje orientado $e_k = v_i \to v_j$: $D_{ki} = -1$, $D_{kj} = 1$ y vale 0 en las demás entradas de la fila k.

En otros términos, una matriz de incidencia dirigida representa una orientación de los ejes de un grafo no dirigido G. Y el espacio de matrices de incidencia dirigida asociado a G equivale al espacio de todas las orientaciones posibles de G.

La *matriz laplaciana*

La *matriz laplaciana* (L) de un grafo G es una matriz de $n \times n$ definida del siguiente modo:

$$L_{ij} = \begin{cases} deg(v_i) & \text{si } i = j \\ -1 & \text{si existe el eje } (v_i, v_j) \\ 0 & \text{en otro caso} \end{cases}$$

El rango de L es $n - c$ donde c es la cantidad de componentes conexas. En particular si G es conexo, el rango de L es $n - 1$

Relación entre las matrices L y D

Es fácil verificar que las matrices L y D se relacionan de acuerdo a la expresión:

$$L = D^t D$$

Es importante notar que L es independiente de cualquier orientación que se le asocie al grafo. Y que el rango de D es el mismo que el de L. Como en nuestro caso G es conexo el rango de D es $n - 1$.

Sistema lineal inducido por D

Recordemos que el problema que queremos resolver es encontrar la matriz de incidencia dirigida D, es decir la "mejor" orientación de los ejes del grafo G, de modo de minimizar la expresión:

$$E(x) = \|D\boldsymbol{x} - \boldsymbol{f}\|^2$$

Para eso será preciso explorar el espacio de matrices de incidencia dirigida asociado a G. Una vez que conseguimos una matriz candidata, el problema de minimización es equivalente a encontrar dónde se anula el gradiente de $E(x)$:

$$\nabla E = \left[\frac{\partial E}{\partial x_1}, \ldots, \frac{\partial E}{\partial x_n}\right] = D^t Dx - D^t f = 0$$

Que a su vez es equivalente a resolver el sistema lineal:

$$D^t Dx = Lx = D^t f$$

Mediante métodos iterativos es posible converger al punto más cercano al vector $D^t f$ contenido en el subespacio generado por L (la proyección ortogonal de $D^t f$). Específicamente en este trabajo emplearemos el método de *Gradiente Conjugado* [Saad, 2007] para resolver el sistema lineal.

Es importante mencionar que en el último tiempo ha habido grandes avances en la creación de nuevos métodos iterativos para resolver esta clase particular de sistemas lineales (aquellos que involucran una matriz laplaciana) denominados *sistemas laplacianos* ([Koutis, Miller, Peng, 2010], [Spielman, 2010], [Spielman, Teng, 2004], [Teng, 2010]).

Norma de la proyección ortogonal

Para elaborar el criterio que nos permita comparar dos matrices de incidencia dirigida y decidir cuál es mejor en el contexto del problema que queremos resolver, vamos a referirnos a nociones básicas de proyección ortogonal de un vector sobre un subespacio de un espacio vectorial, [Ward, Kincaid, 2009].

Sean $v, w \in \mathbb{R}^n$, notamos $p_v(w)$ a la proyección de w sobre v. Mediante relaciones trigonométricas básicas podemos calcular $\|p_v(w)\|$ (la norma de $p_v(w)$):

$$\frac{|vw|}{\|v\|} = \frac{|cos(\alpha)|\ \|v\|\ \|w\|}{\|v\|} = |cos(\alpha)|\ \|w\| = \|p_v(w)\|$$

De modo más general, notamos $p_S(w)$ a la proyección ortogonal de w sobre un subespacio $S \subset \mathbb{R}^n$ generado por una base ortogonal $S = \langle s_1, \ldots, s_k \rangle$. La norma de $p_S(w)$ puede calcularse de modo análogo:

$$\|p_S(w)\| = \| \begin{bmatrix} \|p_{s_1}(w)\| \\ \ldots \\ \|p_{s_k}(w)\| \end{bmatrix} \|$$

Para dar una intuición geométrica del hecho precedente, supongamos (sin pérdida de generalidad) que $S = \langle e_1, \ldots, e_k \rangle$ está dada por los primeros k vectores canónicos y $w = (w_1, \ldots, w_n)$. Claramente la proyección ortogonal de w sobre S es $(w_1, \ldots, w_k)$ (las primeras k componentes de w). Entonces verifiquemos la expresión anterior:

$$\| \begin{bmatrix} \|p_{s_1}(w)\| \\ \ldots \\ \|p_{s_k}(w)\| \end{bmatrix} \| = \| \begin{bmatrix} |cos(\alpha_1)|\ \|w\| \\ \ldots \\ |cos(\alpha_k)|\ \|w\| \end{bmatrix} \| = \| \begin{bmatrix} w_1 \\ \ldots \\ w_k \end{bmatrix} \|$$

Informalmente, el caso general se puede reducir a este caso simple mediante una matriz de rotación, que preserva las normas.

2.2. Metaheurísticas

En el contexto del tratamiento de problemas de complejidad NP, el enfoque metaheurístico ([Gendrau, Potvin, 2010], [Talbi, 2009]) procura encontrar buenas soluciones (no necesariamente las mejores) en tiempo polinomial. Algunas metaheurísticas son:

- Simulated-annealing
- Tabu-search
- GRASP
- Algoritmos genéticos
- Colonia de hormigas

Encontrar la metaheurística más adecuada depende del problema que se quiera resolver y debe ser contrastada empíricamente con los resultados obtenidos por otras metaheurísticas.

Específicamente una metaheurística de tipo *GRASP* combina dos aspectos. Por un lado, requiere cierto tipo de búsqueda local, es decir: explorar un entorno de la solución de la iteración actual, para lo cual es preciso definir una noción de vecindad en el espacio; en nuestro caso se tratará de definir una noción de vecindad en el espacio de matrices de incidencia dirigida asociado a un grafo. Y por otro lado, GRASP requiere cierto grado de azar en la elección del vecino a explorar en la siguiente iteración; esto busca impedir que el algoritmo se estabilice en torno a un mínimo local (no necesariamente un mínimo global).

En la próxima sección se presentará el algoritmo y se describirán las particularidades de la implementación de una metaheurística GRASP en el contexto del problema que queremos resolver.

3. Implementación

En esta sección describiremos el algoritmo y los detalles de la implementación.

3.1. La "mejor" matriz de incidencia dirigida

El primer paso de la estrategia que vamos a desarrollar es tratar de encontrar la "mejor" matriz de incidencia dirigida, o dicho en otros términos: de orientar los ejes del grafo G del mejor modo posible para minimizar el error:

$$E(x) = \|D\boldsymbol{x} - \boldsymbol{f}\|^2$$

Como el espacio de matrices de incidencia dirigida asociado a G es un espacio de tamaño exponencial respecto a la cantidad de ejes, dado que cada eje puede tener dos orientaciones, nuestra propuesta está centrada en construir un algoritmo que devuelva la "mejor solución posible".

Preprocesamiento de las hojas del grafo

Inicialmente proponemos un preprocesamiento del grafo con el objetivo de simplificarlo. En la formulación del problema de minimización:

$$E(x) = \|D\boldsymbol{x} - \boldsymbol{f}\|^2 = \sum_{e \in E} (dx(e) - f(e))^2$$

Consideremos una hoja v de G (un nodo de grado 1), a su único vecino w y al eje que los liga $e = (v, w)$. Dependiendo de la orientación de e, surge que $x(v) = x(w) \pm f(e)$, es decir que $x(v)$ queda determinado por $x(w)$, $f(e)$ y la elección de una orientación de e.

Por lo tanto, es posible reducir el grafo original G, eliminando recursivamente sus hojas de modo de obtener un grafo más simple (un grafo con menos ejes y nodos). Un corolario es que si G es un árbol, entonces $x : V \to \mathbb{R}$ queda determinada por alguna orientación de los ejes y por $x(v)$ para un nodo v.

De ahora en adelante consideraremos que el grado de los nodos de G es al menos 2.

Una base semi-ortogonal

Un hecho simple que se deriva de la expresión que vincula a las matrices L y D:

$$L = D^t D$$

es que el producto interno de dos columnas distintas de una matriz de incidencia dirigida vale -1 o 0 dependiendo de si existe un eje entre los nodos asociados a dichas

columnas o no. Notar el hecho evidente de que si el producto es 0, las columnas son ortogonales.

Ahora notemos lo siguiente: si G es conexo, el ángulo entre dos columnas d_i y d_j de D está contenido en el rango $[\frac{\pi}{2}, \frac{2\pi}{3}]$. Para ver esto, supongamos que existe un eje entre los nodos v_i y v_j asociados a dichas columnas. El ángulo puede medirse de este modo:

$$cos(\alpha)\ \|d_i\|\ \|d_j\| = d_i d_j \implies cos(\alpha) = \frac{d_i d_j}{\|d_i\|\ \|d_j\|}$$

El módulo de esta fracción se maximiza cuando las normas de las columnas son más pequeñas o de modo equivalente: cuando los grados de los nodos correspondientes se minimizan. Respecto a esta última observación, cabe aclarar que la norma de cada columna de D es igual al grado del nodo asociado a esa columna, esto se deduce de la diagonal de $L = D^t D$. Recordemos que el grado de cada nodo de G, luego de preprocesar las hojas, es al menos 2. Entonces resulta que:

$$-\frac{1}{2} \leq cos\ \alpha = \frac{d_i d_j}{\|d_i\|\ \|d_j\|} \leq 0, \quad \forall i \neq j.$$

Implicando $\alpha \in [\frac{\pi}{2}, \frac{2}{3}\pi] \cup [\frac{4}{3}\pi, \frac{3}{2}\pi]$.

Denotaremos base *semi-ortogonal* a una base (de un $\mathbb{R}$-espacio vectorial) tal que el ángulo entre cualquier par de generadores está en el rango $[\frac{\pi}{2}, \frac{2}{3}\pi] \cup [\frac{4}{3}\pi, \frac{3}{2}\pi]$.

El hecho de que las columnas de una matriz de incidencia dirigida definan una base semi-ortogonal va a justificar el criterio que emplearemos para compararlas.

La α-*norma*. Un criterio para comparar matrices de incidencia dirigida

Una interpretación geométrica del problema de minimización que queremos resolver es la siguiente: encontrar la "mejor" matriz de incidencia dirigida D tal que la distancia de f sobre el subespacio generado por las columnas de D sea mínima. En el caso de que el vector f esté contenido en el subespacio, entonces el problema tiene solución exacta, es decir: el sistema lineal $Dx = f$ tiene solución.

Para resolver el problema, necesitamos un criterio para comparar matrices de incidencia dirigida asociadas al grafo G. Como las columnas de una matriz de incidencia dirigida D definen una base semi-ortogonal, aproximaremos la norma de la proyección ortogonal de f sobre D como si las columnas de D fueran ortogonales y denotaremos a esta aproximación α-*norma*. De este modo, entre dos matrices de incidencia dirigida D y $\bar{D}$ nuestro criterio será optar por aquella con mayor α-norma. La justificación de esta decisión se basa en que calcular una base ortonormal del subespacio generado por las columnas de D, mediante por ejemplo el método de *Gram-Schmidt* [Ward, Kincaid, 2009] no es viable en términos de costo computacional.

La α-norma se puede calcular en dos pasos. El paso inicial sería calcular el producto interno de f con las columnas de D:

$$D^t f = \begin{bmatrix} cos(\alpha_1)\ \|d_1\|\ \|f\| \\ \dots \\ cos(\alpha_n)\ \|d_n\|\ \|f\| \end{bmatrix}$$

Y luego dividir cada componente por la norma correspondiente:

$$\alpha(D, f) = \|D^t f \begin{bmatrix} \|d_1\|^{-1} \\ \dots \\ \|d_n\|^{-1} \end{bmatrix}\| = \| \begin{bmatrix} cos(\alpha_1)\ \|f\| \\ \dots \\ cos(\alpha_n)\ \|f\| \end{bmatrix}\| =$$

$$\|f\| \sqrt{cos^2(\alpha_1) + \ldots + cos^2(\alpha_n)}$$

Notemos que para cambiar la dirección de un eje en una matriz de incidencia dirigida D, hay que cambiarle el signo a la fila correspondiente al eje que se quiere modificar. Y de modo más general, si queremos cambiar la dirección de varios ejes simultáneamente la nueva matriz $\bar{D}$ se puede calcular del siguiente modo:

$$\bar{D} = \bar{I} D$$

donde $\bar{I}$ es la matriz diagonal de $m \times m$ que tiene un -1 en cada fila asociada a cada eje que queremos invertir y 1 en el resto de sus componentes.

Notemos que la norma de las columnas de D y $\bar{D}$ son iguales porque sólo difieren en el signo de algunas componentes. Por lo tanto, la norma de las columnas de las matrices de incidencia dirigida puede calcularse una sola vez.

3.2. El enfoque metaheurístico

El problema que queremos resolver se traduce en encontrar una matriz de incidencia dirigida D que maximice la α-norma de f. Dado que el tamaño del espacio de matrices de incidencia dirigida es 2^m (donde $|E| = m$), computar el problema para grafos de tamaño grande es difícil de resolver. Este espacio es extremadamente grande incluso para grafos pequeños. Por lo tanto, el problema admite un enfoque metaheurístico que nos permita encontrar una solución "buena" (aunque no necesariamente la mejor).

Como ya mencionamos, las metaheurísticas *GRASP* combinan azar y búsqueda local. La búsqueda local se refiere a la exploración de una vecindad de un punto del espacio. En nuestro caso el espacio es el conjunto de matrices de incidencia dirigida asociado a G. Una definición obvia de vecindad de una matriz de incidencia dirigida D es el conjunto de matrices que difieren en la dirección de un eje, o sea aquellas que tienen una fila invertida con respecto a D. Cada vez que se encuentra un mínimo (o máximo), lo conservamos. La sencillez de GRASP y la posibilidad de paralelizarlo de un modo natural, lo convierten en una alternativa razonable para nuestro problema.

3.3. El algoritmo

La estrategia que presenta nuestro algoritmo es una variante en la que invertimos los aspectos de azar y búsqueda local de *GRASP*: primero elegimos al azar un conjunto de vecinos y luego definimos como nuevo punto a explorar al mejor de ellos (el que tenga mayor α-norma).

El algoritmo está controlado por dos parámetros:

- La vecindad de un punto está determinada por el parámetro ϕ, que establece la cantidad de ejes del punto que deben modificarse al azar para construir un vecino.
- La cantidad de vecinos del punto que debemos explorar en cada iteración está dada por el parámetro ψ

A continuación presentamos el pseudocódigo del algoritmo y luego comentamos las funciones auxiliares.

Algorithm 1 Integrate(G, f, ϕ, ψ)

```
currentPoint ← Initialize(G)
bestPoint ← currentPoint
while p do
    Ω ← Neighbors(currentPoint)
    bestNeighbor ← BestNeighbor(Ω, f)
    if α(bestNeighbor, f) > α(bestPoint, f) then
        bestPoint ← bestNeighbor
    currentPoint ← bestNeighbor
Solve laplacian system Lx = D^t f
return x
```

La función *Initialize(G)* se encarga de construir una matriz de incidencia dirigida inicial. Concretamente, dado un etiquetado de los nodos, se define la orientación de un eje desde el nodo menor al de mayor etiqueta.

La función *Neighbors(currentPoint)* devuelve un conjunto de vecinos del punto que se está explorando en la iteración. El conjunto de vecinos se arma en base a los parámetros ϕ y ψ. El resultado es un conjunto de ψ vecinos cada uno de los cuales difiere en la dirección de exactamente ϕ ejes (elegidos al azar) con respecto $currentPoint$.

La función $BestNeighbor(\Omega, f)$ devuelve el mejor vecino del conjunto de entrada. O sea, aquel que tiene mayor α-norma. Y por último, la función $\alpha(bestNeighbor, f)$ calcula la α-norma de la matriz de incidencia de entrada.

La condición de corte del $While$ es un predicado (p) que permite terminar la ejecución por criterios distintos: cantidad de iteraciones, tiempo de ejecución transcurrido, margen de error de la mejor solución encontrada dentro de un rango de aceptación.

Y por último, el sistema laplaciano ($Lx = D^t f$) se resuelve mediante el método de *Gradiente Conjugado*.

4. Resultados

En esta sección describimos los resultados obtenidos. Primero comentamos el criterio que adoptamos para determinar los valores de los parámetros ϕ y ψ. Y posteriormente nos enfocamos en el análisis del algoritmo evaluado sobre tres familias de grafos:

- *Grafos completos*: son aquellos en los que todo par de nodos está ligado por un eje.
- *Ciclos simples*: grafo que define un ciclo; se puede pensar en un polígono simple como modelo geométrico.
- *Mallas poligonales*: son representaciones de superficies inmersas en el espacio tridimensional; tienen más estructura que un grafo porque los nodos tienen asociada una posición en el espacio.

La elección de las dos primeras familias está justificada por el hecho de que los grafos completos y los ciclos simples determinan los casos extremos en términos del grado de sus nodos: los grafos completos maximizan el grado, mientras que los ciclos simples lo minimizan. Recordar que el grado de los nodos es por los menos 2. Y los grafos asociados a mallas poligonales son aquellos en los que inicialmente se planteó el problema que analizamos en este trabajo.

Es importante notar que las instancias de input del algoritmo (G, f) donde $G = (V, E)$ es un grafo y $f : E \to \mathbb{R}$ es una función de pesos, se construyeron de modo tal de que exista una solución exacta. Es decir, se eligió de manera aleatoria (distribuida uniformemente en un intervalo específico) una función de pesos en los nodos $x : V \to \mathbb{R}$ cuyo diferencial es f. En este contexto medimos el error del siguiente modo:

$$error(D) = \frac{\|Dx - f\|}{\|f\|}$$

donde D es la matriz de incidencia dirigida y x es la función de pesos en los nodos que devuelve el algoritmo. Por lo tanto, el error está contenido en el intervalo $[0, 1]$

Finalmente, es importante mencionar que los experimentos se limitaron a evaluar la convergencia de cada instancia de input. Es decir, a exponer cómo disminuye el error respecto a la cantidad de iteraciones.

4.1. Ajuste de parámetros

Recordemos que los parámetros del algoritmo, ϕ y ψ denotan respectivamente la cantidad exacta de ejes en los que difieren los vecinos de la matriz de incidencia dirigida (de la iteración actual) y el tamaño del conjunto de vecinos.

Para estimar ϕ y ψ empleamos un grafo completo de 100 nodos y consideramos distintos valores posibles de los parámetros: $\phi \in \{1, 2, 3, 4\}$ y $\psi \in \{2, 4, 6, 8, 10\}$. Cada par (ϕ_i, ψ_j) define un test. Para cada test el algoritmo se ejecutó durante 10^4 iteraciones y se repitió 5 veces. El resultado del error medio está reflejado en la tabla 1 y en el heat-map de la figura 1. Lo que se confirma inmediatamente es algo razonable: cuanto mayor es el tamaño del conjunto de vecinos (ψ), menor es el error del algoritmo. Por otro lado, surge que la mejor elección de ϕ es 2, o sea conviene considerar vecinos que difieran en exactamente 2 ejes. Esto último, se pudo confirmar en una prueba más exhaustiva en la cual fijamos ψ en 10 y ejecutamos 50 veces cada test durante $2{,}10^4$ iteraciones. El resultado está reflejado en el boxplot de la figura 2. En este caso, es interesante notar cómo disminuye la varianza a medida que ϕ aumenta.

En base a estos resultados y al tamaño de las instancias que utilizamos en los experimentos, el criterio que adoptamos para ajustar los parámetros es el siguiente:

- $\phi = 2$
- $\psi = 10^{-2}|E|$: esta proporción de vecinos permite procesar instancias de grafos de mayor tamaño.

Cuadro 1. AJUSTE DE PARÁMETROS

Parámetros		
ϕ	ψ	error promedio
1	2	0,5189745
1	4	0,2568497
1	6	0,2129859
1	8	0,1980620
1	10	0,1767993
2	2	0,5261091
2	4	0,3491206
2	6	0,2365749
2	8	0,2093766
2	10	0,1237789
3	2	0,5261091
3	4	0,4260077
3	6	0,2974878
3	8	0,2450824
3	10	0,2033154
4	2	0,5260864
4	4	0,5029136
4	6	0,3957194
4	8	0,3118229
4	10	0,2619005

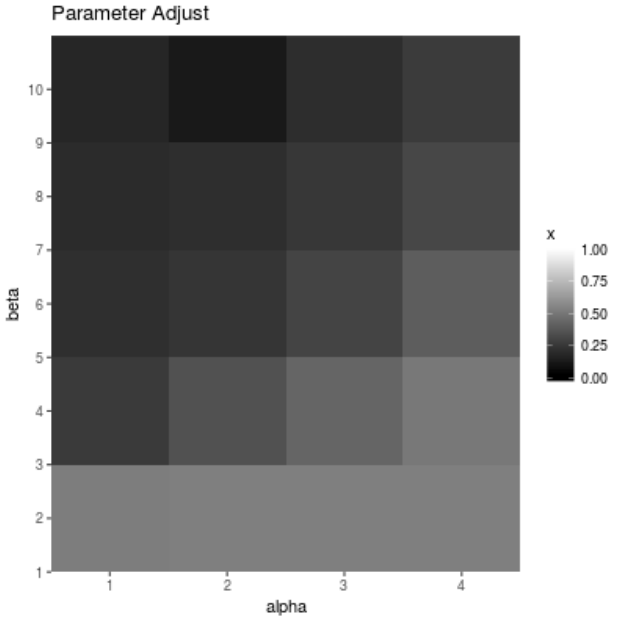

Figura 1. Ajuste de parámetros

Tenemos que aclarar que si bien el criterio no es exhaustivo, estos son valores de referencia para evaluar el algoritmo en condiciones uniformes.

4.2. Grafos completos

Los grafos completos son aquellos en los que todo par de nodos está conectado por un eje. Esta familia es importante en el contexto del problema que estamos analizando porque a medida que crece el tamaño de la instancia del grafo de input, la α-*norma* estima mejor a la proyección ortogonal de f sobre el subespacio generado por las columnas de las matrices de incidencia dirigida. Más específicamente, el ángulo entre dos columnas de una matriz de incidencia dirigida puede calcularse mediante la expresión:

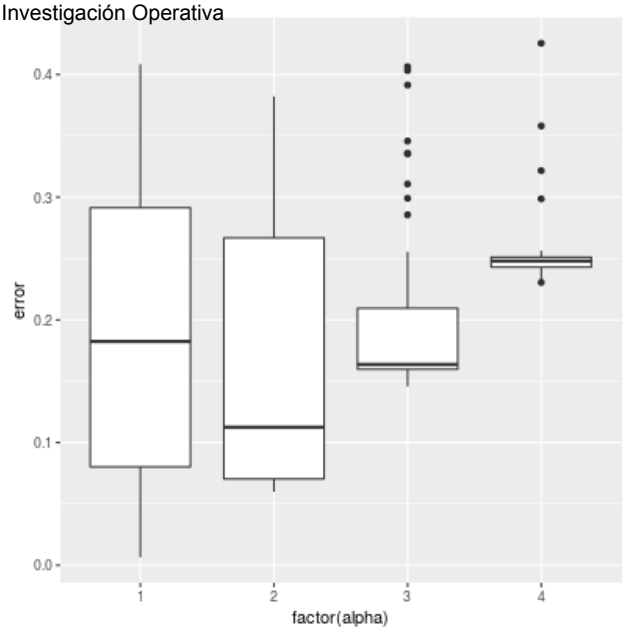

Figura 2. Ajuste de parámetros

$$cos(d_i, d_j) = \frac{d_i d_j}{\|d_i\| \ \|d_j\|} = \frac{d_i d_j}{deg(v_i) \ deg(v_j)}$$

donde $cos(d_i, d_j)$ es el coseno del ángulo entre las columnas de D correspondientes a los nodos v_i y v_j del grafo. A medida que el grado de los nodos crece, el coseno tiende a 0, o sea: las columnas de D se van volviendo más ortogonales.

En este caso, consideramos tres tipos de instancias de grafos completos: de 100, 200 y 400 nodos. Para cada tipo de instancia, se hicieron 20 experimentos, cada uno de los cuales se ejecutó durante $2{,}10^4$ iteraciones. Es importante notar que la cantidad de ejes de un grafo completo es del orden de n^2 (donde $|V| = n$). La figura 3 muestra la media del error en función de la cantidad de iteraciones para cada tipo de instancia. Se observa una velocidad de convergencia aceptable.

4.3. Ciclos simples

Los grafos que definen ciclos simples también representan una familia de interés. En este caso, como cada nodo tiene solamente dos vecinos, la columna de una matriz de incidencia dirigida asociada a un cierto nodo, es ortogonal a todas las demás columnas excepto a las asociadas a los dos nodos vecinos. En ese caso, el ángulo que forman dos columnas asociadas a dos nodos vecinos es $\frac{2\pi}{3}$. En esta familia de grafos la cantidad de ejes es del orden de $|V| = n$.

En este caso consideramos tres tipos de instancias de ciclos simples: de 1000, 2000 y 4000 nodos. Para cada tipo de instancia, se hicieron 20 experimentos, cada uno de los cuales se ejecutó durante 500 iteraciones. La figura 4 muestra la media del error en función de la cantidad de iteraciones para cada tipo de instancia. También en este caso, se observa una velocidad de convergencia aceptable.

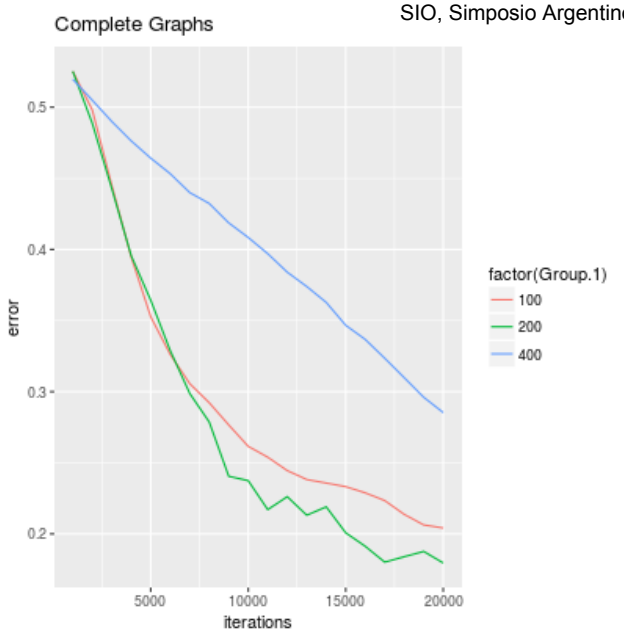

Figura 3. Grafos completos

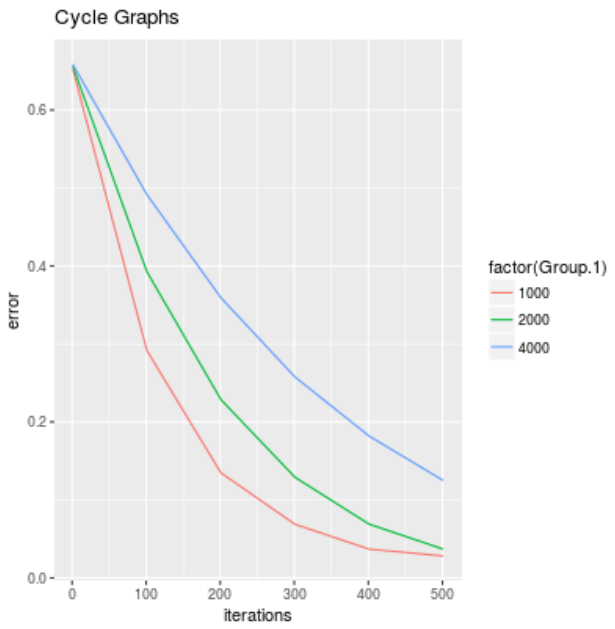

Figura 4. Ciclos simples

Es importante mencionar que en este caso, la limitación que existe para procesar instancias con una mayor cantidad de ejes es la resolución del sistema laplaciano $Lx = D^t f$. La matriz L es de tamaño $n \times n$, eso impone un límite a esta implementación. Una implementación paralelizada de la resolución del sistema lineal permitiría que el algoritmo escale sin limitaciones.

4.4. Mallas poligonales

Recordemos que las *mallas poligonales* permiten representar superficies inmersas en el espacio tridimensional. Las caras son aproximaciones lineales de pequeñas vecindades de la superficie. Y el grafo subyacente de una malla describe los bordes de las caras.

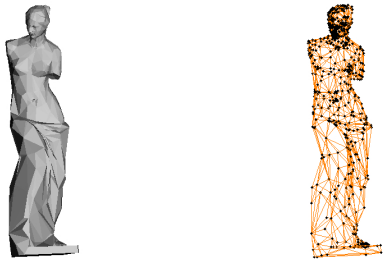

Figura 5. **venus**: malla poligonal y su grafo subyacente

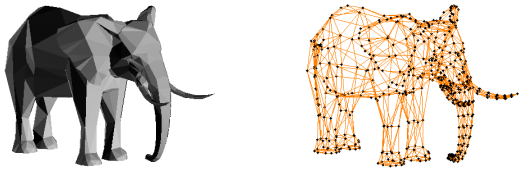

Figura 6. **elephant**: malla poligonal y su grafo subyacente

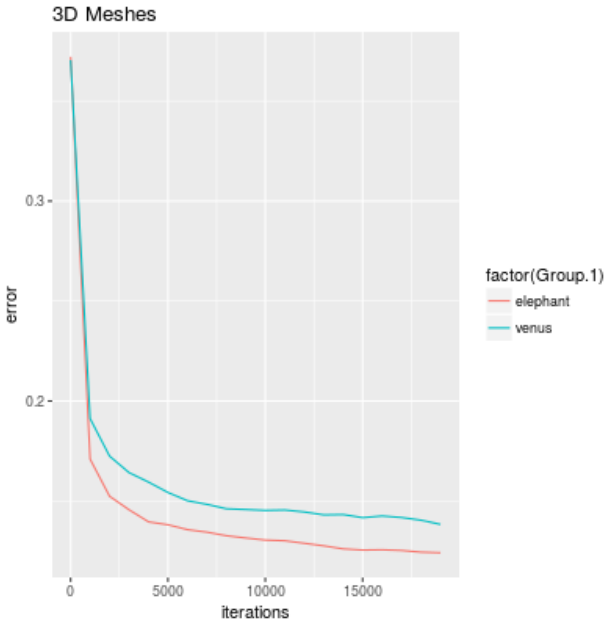

Figura 7. Mallas poligonales

Cuadro 2. MALLAS POLIGONALES

malla	$\|V\|$	$\|E\|$
venus	819	2452
elephant	623	1759

En este caso empleamos dos superficies compactas para evaluar el algoritmo: *venus* (figura 5) y *elephant* (figura 6). La tabla 2 detalla la información de los grafos ($G = (V, E)$) de ambas mallas.

Para cada malla se hicieron 20 experimentos, cada uno de los cuales se ejecutó durante $2{,}10^4$ iteraciones. La figura 7 muestra la media del error en función de la cantidad de iteraciones. En estos casos observamos que inicialmente la convergencia es muy rápida, pero a partir de un cierto punto se estabiliza. Este hecho da indicios de que esta estrategia debería complementarse con otra que permita refinar la mejor solución encontrada.

5. Conclusiones y Trabajo futuro

En este trabajo abordamos un problema que denominamos *integración en grafos* que "integra" (en términos analíticos) un dato diferencial en los ejes. Este enfoque permite resolver distintos tipos de problemas diferenciales que se modelen con grafos. El algoritmo metaheurístico que planteamos está basado en comparar *matrices de incidencia dirigida* asociadas al grafo de input. Para ello consideramos un criterio que denominamos α*-norma* vinculado a la proyección ortogonal del vector que representa al dato diferencial sobre el subespacio generado por las columnas de las matrices de incidencia dirigida.

Evaluamos el algoritmo en instancias de tres familias de grafos: grafos completos, ciclos simples y mallas poligonales. Y observamos que se comporta adecuadamente dado que converge a una solución aproximada del problema en un tiempo razonable. No obstante, hay que remarcar que para ciertas instancias de input, a partir de cierto punto el algoritmo se estabiliza entorno a un error de $0{,}1$; esto sugiere que la técnica que presentamos en este trabajo debería complementarse con otra para refinar la solución.

El algoritmo es simple, por lo cual es posible paralelizarlo de modo horizontal y procesar instancias de mayor tamaño mediante técnicas de procesamiento distribuido (ej.: *map-reduce*).

En la implementación que presentamos, la mayor limitación está relacionada a la resolución de *sistemas laplacianos*, es decir: sistemas lineales $Lx = b$ en los que L es una *matriz laplaciana*. Actualmente hay importantes líneas de trabajo abordando nuevas metodologías iterativas que reducen el tiempo de procesamiento de este tipo de sistemas. En una futura implementación sería importante incorporar estos enfoques.

En un trabajo futuro, planeamos formalizar adecuadamente los argumentos probabilísticos por los cuales la α-norma es una aproximación razonable de la norma de la proyección ortogonal de un vector sobre un subespacio. También planeamos estudiar alternativas a la función *Initialize(G)* de modo de poder reducir el tiempo de computo del algoritmo.

Respecto a la complejidad del problema planteado, sospechamos que es NP-completo (lo que justifica la utilización de metaheurísticas) y será el tema de estudio en un trabajo futuro. Respecto a esto, uno de los objetivos será plantear el problema en el contexto de programación entera cuadrática para calcular soluciones exactas a instancias más pequeñas y compararlas con nuestra estrategia. Notar que la α-norma puede pensarse como una ecuación cuadrática sobre el espacio de matrices D y esto es un caso particular de lo estudiado en [Manders, Adleman, 1976].

Agradecimientos

Queremos agradecer al proyecto UndavCyT 2014 "Especificaciones formales tempranas del comportamiento de sistemas de software".

Los autores queremos agradecer al Prof. Gabriel Taubin por plantearnos el problema de integración de formas diferenciales en el contexto de mallas poligonales y por su generosa disposición a responder nuestras preguntas. Asimismo queremos agradecer al Prof. Fernando Cukierman por guiar nuestro estudio en el área de geometría diferencial.

Agradecemos a los revisores del Simposio de Investigación Operativa por sus valiosos aportes para enriquecer este trabajo.

Referencias

[Taubin et al., 2002] Bernardini F., Martin I., Mittleman J., Rushmeier H., Taubin G., *Building a Digital Model of Michelangelo's Florentine Pieta'* IEEE Computer Graphics and Applications.

[Botsch, Kobbelt, Pauly, Alliez, Lévy, 2010] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B., *Polygon Mesh Processing* A. K. Peters, 2010.

[Crispell, Mundy, Taubin, 2008] Crispell D., Mundy J., Taubin G., *Parallax-Free Aerial Video Registration* British Machine Vision Conference 2008.

[Desbrun, Kanso, Tong, 2006] Desbrun M., Kanso E., Tong Y., *Discrete differential forms for computational modeling* Proceeding SIGGRAPH '06 ACM SIGGRAPH 2006 Courses Pages 39-54 .

[Gendrau, Potvin, 2010] Gendreau M. and Potvin J., *Handbook of Metaheuristics*, Springer, 2010.

[Harari, 1969] Harary F., *Graph Theory*, Addison-Wesley, 1969.

[Hatcher, 2004] Hatcher A., *Algebraic Topology*, Cambridge University Press.

[Koutis, Miller, Peng, 2010] Koutis I., Miller G. L. and Peng R., *Approaching optimality for solving sdd linear systems*, 51st Annual Symposium on Foundations of Computer Science, IEEE, 2010, pp. 235-244.

[Lanman, Raskar, Agrawal, Taubin, 2008] Lanman D., Raskar R., Agrawal A., Taubin G., *Shield Fields: Modeling and Capturing 3D Occluders*, ACM Transactions on Graphics (TOG), Vol 27, Issue 5, December 2008, Siggraph Asia Proceedings.

[Manders, Adleman, 1976] Manders K., Adleman L., *NP-complete decision problems for quadratic polynomials*, Proceeding STOC '76 Proceedings of the eighth annual ACM symposium on Theory of computing pp 23-29.

[REVEAL] Reconstruction and ExploratoryVisualization: Engineering meets Art / ArchaeoLogy.

[Ronfard, Taubin, 2007] Ronfard R., Taubin G., *Introducing 3D Cinematography*, IEEE Computer Graphics and Applications (pp. 18-20).

[Saad, 2007] Saad Y., *Iterative methods for sparse linear systems*, SIAM, 2007.

[Spielman, 2010] Spielman D. A., *Algorithms, graph theory, and linear equations in laplacian matrices*, Proceedings of the International Congress of Mathematicians, vol. 4, 2010, pp. 2698-2722.

[Spielman, Teng, 2004] Spielman D. A. and Shang-Hua Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, Proceedings of the thirty-sixth annual ACM symposium on Theory of Computing, ACM, 2004, pp. 81-90.

[Spivak, 1965] Spivak M., *Calculus on Manifolds*, W. A. Benjamin.

[Talbi, 2009] Talbi E.G., *Metaheuristics: from design to implementation*, Wiley, 2009.

[Taubin, Cooper, 1992] Taubin G., Cooper D.B., *2D and 3D Object Recognition and Positioning with Algebraic Invariants and Covariants*, Academic Press.

[Teng, 2010] Shang-Hua Teng, *The laplacian paradigm: Emerging algorithms for massive graphs*, Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol. 6108, 2010, pp. 2-14.

[Tu, 2008] Tu, L. W. *An Introduction to Manifolds*, Springer.

[Ward, Kincaid, 2009] Ward C., Kincaid D., *Linear Algebra: Theory and Applications* Sudbury, Ma: Jones and Bartlett. pp. 544, 558.

[Zhao, Taubin, 2009] Zhao Y., Taubin G., *Real-Time High Definition Stereo on GPGPU using Progressive Multi-Resolution Adaptive Windows* NVIDIA Research Summit 2009, NVIDIA GPU Technology Conference.