Rev. Int. Mét. Num. Cálc. Dis. Ing.
Vol. **19, 1,** 89-109 (2003)

Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Estructuras de datos geométricas para algoritmos de refinamiento basados en el esqueleto

José Pablo Suárez

Departamento de Cartografía y Expresión Gráfica en la Ingeniería
Universidad de Las Palmas de Gran Canaria
Edificio de Ingenierías, Campus de Tafira
35017 Las Palmas de Gran Canaria, España
Tel.: 34-928-45 72 69; Fax: 34-928-45 18 72
e-mail: jsuarez@dcegi.ulpgc.es

Ángel Plaza

Departamento de Matemáticas
Universidad de Las Palmas de Gran Canaria
Edificio de Matemáticas e Informática, Campus de Tafira
35017 Las Palmas de Gran Canaria, España
Tel.: 34-928-45 88 27; Fax: 34-928-45 87 11
e-mail: aplaza@dma.ulpgc.es

Graham F. Carey

Texas Institute for Computational and Applied Mathematics (TICAM)
ASE/EM Dept., University of Texas at Austin
201 E. 24th Street, ACE 6.412, Austin, Texas 78712, EE.UU.
Tel.: 1-512-471 33 12; Fax: 1-512-471 86 94
e-mail: carey@cfdlab.ae.utexas.edu

Resumen

En este artículo se presenta y discute una clase de algoritmos de refinamiento adaptativo para generar mallas de triángulos y tetraedros no estructuradas en dos y tres dimensiones. Concretamente, se estudian los algoritmos de refinamiento basados en el esqueleto (Skeleton Based Refinement (SBR) algorithms) propuestos por Plaza y Carey[23] y se presenta una versión que hace uso del grafo del esqueleto de las mallas triangulares. Mediante el uso de estas estructuras de datos derivadas del concepto de esqueleto de la triangulación se reformulan estos algoritmos y adquieren una descripción más natural y consistente. El caso bidimensional es discutido con detalle y para el caso 3D se propone una nueva estructura de datos tipo grafo basada en las caras triangulares de los tetraedros. Se muestran experimentos en 2D y se exploran algunas propiedades asociadas al grafo.

Palabras clave: *refinamiento, esqueleto, estructuras de datos, bisección por la arista.*

GEOMETRIC DATA STRUCTURES FOR REFINEMENT BASED ON SKELETON

Summary

In this paper we discuss a class of adaptive refinement algorithms for generating unstructured meshes in two and three dimensions. We focus on Skeleton Based Refinement (SBR) algorithms as proposed by Plaza and Carey[23] and provide an extension that involves the introduction of the skeleton graph for meshes consisting of simplex cells. By the use of data structures derived from the skeleton graph, we reformulate the Skeleton Based Refinement scheme and derive a more natural and consistent approach for this class of adaptive refinement algorithms. We discuss in detail the graphs for 2D refinement of triangulations, while for the 3D we propose a corresponding new face-based data structure for tetrahedra. Experiments using the two dimensional algorithm and exploring the properties of the associated graph are provided.

Keywords: *refinement, skeleton, data structure, edge bisection.*

 ISSN: 0213–1315 Recibido: Mayo 2001

INTRODUCCIÓN

Los algoritmos de refinamiento y desrefinamiento y las estructuras de datos asociadas son parte esencial de las estrategias de adaptación de mallas para lograr cálculos eficientes y fiables para resolver problemas definidos por ecuaciones diferenciales en derivadas parciales. Más aún, como la solución sobre una malla dada nos proporciona un buen punto de partida para resolver iterativamente un problema evolutivo de forma adaptativa, las estrategias de refinamiento y desrefinamiento adaptativo han merecido un creciente interés, así como las estructuras de datos asociadas al uso de métodos multimalla en mallas encajadas. Para un tratamiento general de mallas computacionales y temas afines como generación de mallas y estrategias de solución véase por ejemplo Carey[6]. Los algoritmos de refinamiento y desrefinamiento se pueden usar también para otros propósitos como Minería de Datos, Visualización Computacional, etc.[7,8].

Un punto fundamental de estos algoritmos es la capacidad de modificar la malla. La complejidad de las mallas que pueden usarse demanda algoritmos robustos y eficientes acompañados de estructuras de datos con bajo coste de almacenamiento y bajo coste de actualización. Algunas características de las estructuras de datos necesarias para códigos de elementos finitos han sido descritas, por ejemplo: la del proceso de búsqueda durante la creación de la malla[5,18], el refinamiento de mallas ya existentes[9,13,16] y el proceso de solución del sistema de ecuaciones asociado[18,19].

La estructura de datos es también fundamental en las aplicaciones donde aparecen mallas complejas como por ejemplo en el modelado de sólidos. Una revisión de estas estructuras de datos se puede encontrar en la referencia 32. Existen también algunas librerías de algoritmos y estructuras de datos que pertenecen al área de la Geometría Computacional. Entre estas librerías está la CGAL[14] (Computational Geometry Algorithms Library). CGAL hace uso del paradigma de la programación genérica orientada a objetos para implementar estructuras de datos de tipo combinatorio. Otra librería similar es LEDA[14] (Library of Efficient Data Types and Algorithms). En esta última se puede extraer una estructura de datos general para grafos y mapas planares.

El concepto de esqueleto de una malla nos proporciona un conjunto jerárquico de componentes simpliciales que define la malla de partida en forma recursiva a medida que desciende la dimensión. De esta forma, por ejemplo una malla de tetraedros se puede definir usando el esqueleto, formado por las caras triangulares de los tetraedros que forman la malla inicial, los triángulos se pueden definir por las aristas, que a su vez quedan definidas por el conjunto de nodos de la malla. En la referencia 23 Plaza y Carey presentaron algoritmos de refinamiento basados en el concepto de esqueleto en dimensiones dos y tres y los denominaron 2D-SBR (Two-Dimensional Skeleton Based Refinement Algorithm) y 3D-SBR respectivamente. En ellos se aprovecha la disminución en dimensión que conlleva considerar el esqueleto de una triangulación en lugar de la triangulación misma. Como se decía allí, se puede usar un grafo basado en las aristas para modelar la bisección por la arista mayor. Por ejemplo en el caso bidimensional, usando únicamente la clasificación de las aristas de un triángulo como una arista *mayor* y dos aristas *no-mayores*, se construye un grafo orientado. De forma análoga se puede pensar un grafo en tres dimensiones para explicar distintas clases de tetraedros en el proceso de subdivisión. En la referencia 26 se presentó una estructura de datos adecuada para el algoritmo de refinamiento en 2D basado en el esqueleto. Se mostraban también algunas propiedades como su efectividad y la relación de esta estructura de datos con el concepto del *LEPP* (Longest Edge Propagation Path) definido por Rivara[31]. Como consecuencia se demostraba la finitud del algoritmo 2D-SBR y del esquema 4-T de Rivara.

La teoría de grafos es particularmente útil para modelar problemas que manifiestan relaciones entre objetos. En este artículo se presenta el grafo del esqueleto que explícitamente

es usado en las estructuras de datos de los algoritmos de refinamiento basados en el esqueleto. Estas estructuras de datos surgen de forma natural y son consistentes con esta clase de algoritmos de refinamiento y desrefinamiento.

El artículo está organizado como sigue: primero se resume la notación empleada y se introducen algunas definiciones básicas. Después se presenta una breve revisión de los principales algoritmos de refinamiento y las particiones asociadas en 2 y 3 dimensiones. Se desarrolla a continuación el grafo basado en el esqueleto de una triangulación n-dimensional para $n = 2, 3$ así como realizaciones específicas de los grafos para los algoritmos de bisección basados en la arista mayor como por ejemplo los algoritmos de refinamiento basados en el esqueleto. Se dan nuevas versiones de los algoritmos de refinamiento basados en el esqueleto en 2D y 3D mediante el uso del grafo del esqueleto. Por último, para ilustrar el uso y comportamiento de estos algoritmos se presentan algunos resultados en 2D.

DEFINICIONES

Definición 1 *(m-símplice) Sea* $V=\{X_1, X_2, \cdots, X_{m+1}\}$ *un conjunto de m+1 puntos de* R^n, $m \leq n$, *tal que* $\{\overrightarrow{X_1X_i} : 2 \leq i \leq m+1\}$ *es un conjunto de vectores linealmente independientes. Entonces la clausura convexa de* V *que escribiremos* $S =< V >=< X_1, X_2, \ldots X_{m+1} >$ *se llama m-símplice en* R^n, *los puntos* $X_1, X_2, \ldots, X_{m+1}$ *se llaman vértices de* S *y el número* m *es la dimensión de* S.

Si $m = n$ un n-símplice S se llama sencillamente *símplice* o *triángulo de* R^n; los 2- y 3-símplices se llaman triángulos y tetraedros respectivamente, como es habitual.

Definición 2 *(k-cara) Sea* $S = \langle X_1, X_2, \ldots, X_{n+1}\rangle$ *un n-símplice en* R^n. *Un k-símplice* $T = \langle Y_1, Y_2, \ldots, Y_{k+1}\rangle$ *se llama k-subsímplice o una k-cara de* S *si existen índices* $1 \leq i_1 < i_2 < \ldots < i_{k+1} \leq n+1$ *tales que* $Y_j = X_{i_j}$ *para* $1 \leq j \leq k+1$.

Las 0- y 1-caras de S son precisamente sus vértices y aristas respectivamente.

Definición 3 *(Triangulación) Sea* Ω *un conjunto acotado cualquiera de* R^2, *o* R^3 *con interior no vacío* $\overset{o}{\Omega} \neq \emptyset$ *y borde poligonal* $\partial\Omega$ *y consideremos una partición de* Ω *en un conjunto de triángulos* $\tau = \{t_1, t_2, t_3, \ldots, t_n\}$. *Entonces se dice que* τ *es una triangulación si* $\Omega = \cup t_i$; $int\,(t_i) \neq \emptyset, \forall t_i \in \Omega$ *e* $int\,(t_i) \cap int\,(t_j) = \emptyset$ *si* $i \neq j$.

Una triangulación se dice *conforme* si todo par de símplices adyacentes comparten o una cara entera o una arista entera o un vértice común.

Definición 4 *(Esqueleto) Sea* τ *una malla triangular n-dimensional (*$n = 2$ *o 3). El k-esqueleto de* τ *es la unión de sus k-caras. El* $(n-1)$*-esqueleto se llama también esqueleto.*

Por ejemplo, el esqueleto de una triangulación en dimensión tres está compuesto por las caras de los tetraedros y en dimensión dos el esqueleto es el conjunto de las aristas de los triángulos. El esqueleto se puede entender a su vez como una nueva triangulación. Además si τ es una triangulación conforme en 3D, entonces $skt(\tau)$ es también conforme.

Dos triangulaciones conformes τ y τ^* del mismo conjunto acotado Ω se llaman *anidadas o encajadas* y se escribe $\tau \prec \tau^*$ si se cumple la siguiente condición: $\forall t \in \tau$, $\exists$ t_1, t_2, ..., $t_p \in \tau^*$ tal que $t = t_{1\cup}\ t_2 \ldots \cup t_p$. También se dice que τ es más grosera que τ^* o que τ^* es más fina que τ.

El *camino de propagación por el lado mayor, LEPP (Longest Edge Propagation Path)*, de un triángulo t_0 es definido por Rivara[31] como la lista ordenada de todos los triángulos

adyacentes $\{t_0,t_1,\ldots,t_n\}$, tal que t_i es el triángulo vecino de t_{i-1} por la arista mayor de t_{i-1}. Nótese que al calcular el *LEPP* de un triángulo dado t, t pertenece también a su *LEPP*.

A continuación se recogen algunos conceptos y definiciones de la teoría de grafos usados más adelante[11,12].

Un *grafo geométrico* es un par *(P, L)* donde P es un conjunto no vacío de *nodos* y L es un conjunto (posiblemente vacío) de *enlaces*. El grafo se denota como *G(P, L)* o simplemente *G*. Escribimos $l_k\sim$ *(*p_i*,* p_j*)* para representar el enlace l_k asociado a los nodos p_i, p_j. Los nodos representan objetos y los enlaces relaciones entre esos objetos. Los enlaces pueden tener *etiquetas* o *pesos* y los nodos pueden tener *nombres*. Un subgrafo *G(P', L')* es un grafo tal que $P' \subset P$ y $L' \subset L$. Un *enlace* $l_k\sim$ *(*p_i*,* p_j*)* entre los nodos p_i y p_j se dice que es *incidente* con los nodos p_i y p_j. Los nodos p_i y p_j se llaman *extremos* del enlace (p_i, p_j). Dos enlaces son *adyacentes* si tienen un nodo común. El *grado* de un nodo p_i es el número de enlaces incidentes con p_i. Un grafo *planar* es un grafo que se puede dibujar en el plano conteniendo todos sus enlaces de forma que los enlaces sólo se cortan en los nodos del grafo. Un grafo es *conectado o conexo* si todo par de nodos en G están conectados, en otro caso se dice que es *desconectado*. Una *componente* es un subgrafo conectado maximal. Si los enlaces del grafo son ordenados, entonces el grafo es *dirigido*, y si no lo son, se llama *no dirigido*. Para un grafo dado, si podemos recorrerlo de un nodo a otro por sucesivos enlaces adyacentes sin pasar dos veces por el mismo nodo, se llama una *cadena* en un grafo no dirigido y un *camino* en un grafo dirigido. El *tamaño* de un camino o una cadena es igual al número de sus enlaces. Si una cadena o un camino retorna al nodo inicial, se llama *circuito* o *ciclo* respectivamente. La *distancia* entre dos nodos es la longitud del camino más corto que los une. Recorrer un grafo consiste en visitar todos sus vértices. El recorrido en profundidad (*depth-first traversal*) de un grafo consiste en, partiendo de un nodo inicial, visitar cada nodo adyacente y después cada sucesor de los visitados anteriormente y así sucesivamente.

ALGUNOS ALGORITMOS DE REFINAMIENTO Y PARTICIONES ASOCIADAS

En esta sección se presentan algunas de las particiones más comunes de triángulos y tetraedros usadas en la práctica. Estas particiones constituyen el núcleo de los algoritmos de refinamiento asociados.

Algoritmos de refinamiento y particiones asociadas en 2D

Definición 5 *(Partición semejante 4T) El triángulo original es dividido en cuatro triángulos al conectar los puntos medios de sus aristas por segmentos rectos paralelos a los lados (Figura 1a). En consecuencia, todos los subtriángulos formados son semejantes al original.*

Carey[10] fue el primero en aplicar esta división en el contexto del Refinamiento Adaptativo de Mallas (AMR). Esta partición es una de las más sencillas particiones de triángulos de las que aparecen en la literatura[3]. Bank *et al.*[1] han desarrollado un algoritmo basado en este esquema: los triángulos inicialmente marcados para refinar se dividen mediante la partición semejante 4T (esta partición regular es llamada *red partition).* Para asegurar la conformidad de la malla, se aplica a un conjunto adicional de triángulos unos patrones de refinamiento irregular (*green partition*) mediante la conexión de los nodos en los puntos medios de las aristas con los vértices opuestos.

Definición 6 *(Bisección simple y bisección LE) La bisección simple consiste en dividir el triángulo en dos subtriángulos uniendo el punto medio de una de sus aristas con el vértice*

opuesto (Figura 1b). Si se elige la arista mayor para bisectar el triángulo, se dice que se ha realizado una bisección por el lado mayor o bisección LE.

Definición 7 *(Partición 4T-LE) La partición 4T Longest Edge divide el triángulo en cuatro triángulos, como muestra la Figura 1c, donde el triángulo es primero dividido por su arista mayor y después los dos triángulos que han aparecido son de nuevo divididos uniendo el nuevo nodo con los puntos medios de las dos aristas que quedan.*

La Figura 1d muestra la *partición baricéntrica* de un triángulo t que consiste en conectar el baricentro de t con los vértices y los puntos medios de t.

Rivara ha desarrollado un algoritmo de refinamiento efectivo basado en la bisección por la arista mayor[28,29]. Un algoritmo similar que trabaja con el esqueleto ha sido desarrollado por Plaza y Carey[23]. Ambos algoritmos son equivalentes en el sentido de que producen las mismas triangulaciones.

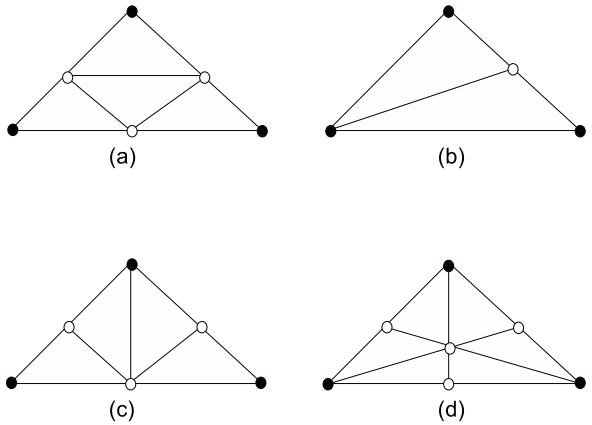


Figura 1. Diversas formas de dividir un triángulo: (a) partición semejante 4T, (b) bisección, (c) partición 4T-LE, (d) partición baricéntrica

Algoritmos de refinamiento y particiones asociadas en 3D

Algunas técnicas basadas en la bisección de tetraedros se han desarrollado en los últimos años. Algoritmos basados en una bisección *simple* por la arista mayor han sido desarrollados por Rivara y Levin[30] y por Muthukrishnan *et al.*[20]. Como cada tetraedro es dividido por su arista mayor hasta que se alcanza la conformidad de la malla, *a priori* no se sabe en cuántos tetraedros va a quedar dividido cada uno de la malla original.

Plaza y Carey[23] han presentado una generalización del algoritmo 4T-LE de Rivara a tres dimensiones. Esta extensión ha sido llamada algoritmo 3D-SBR (3D-Skeleton Based Refinement). Algunas propiedades geométricas de este algoritmo han sido señaladas por Rivara y Plaza [27]. El algoritmo trabaja primero sobre las caras triangulares de los tetraedros, el 2-esqueleto de la teselación en 3D, y después subdivide el interior de cada tetraedro de forma congruente con la subdivisión del esqueleto. Esta idea podría ser aplicada para desarrollar algoritmos semejantes en dimensiones mayores. El algoritmo se puede utilizar sobre cualquier malla inicial de tetraedros sin ningún preproceso. Esta es una propiedad importante con relación a otros algoritmos similares. Como en el caso del refinamiento, también el algoritmo *inverso*, de desrefinamiento, está basado en el 2-esqueleto al asegurar la conformidad de la malla. Después, para reconstruir los tetraedros se utilizan los mismos patrones que fueron usados al refinar[24,25].

Los algoritmos de Bänsch[2] y Liu y Joe[17] dividen cada tetraedro en ocho subtetraedros mediante sucesivas divisiones de las aristas. Kossaczký[15] también ha propuesto un algoritmo de refinamiento que es recursivo. Mukherjee[21] ha presentado un algoritmo equivalente. Sin embargo, todos estos algoritmos, aunque producen las mismas mallas, no se pueden aplicar a cualquier malla inicial y necesitan algún tipo de pre-proceso.

Las particiones de tetraedros análogas a las particiones de la Figura 1a,c están en las Figuras 2 y 3. En cada caso, a la izquierda aparece un ejemplo de tetraedro ilustrando la partición que se ha usado y a la derecha la representación en la que se han cortado las caras a lo largo de tres aristas coincidentes en el vértice D y llevadas al plano. Además, con el fin de ver uniformemente las caras a la izquierda, se aplica una sencilla transformación topológica que deja todas las caras del tetraedro triángulos equiláteros. La idea de representar un tetraedro abierto ha sido utilizada por Bänsch[2] para explicar su algoritmo de refinamiento y muestra de forma sencilla la construcción del grafo en 3D desde el punto de vista de la subdivisión de las caras.

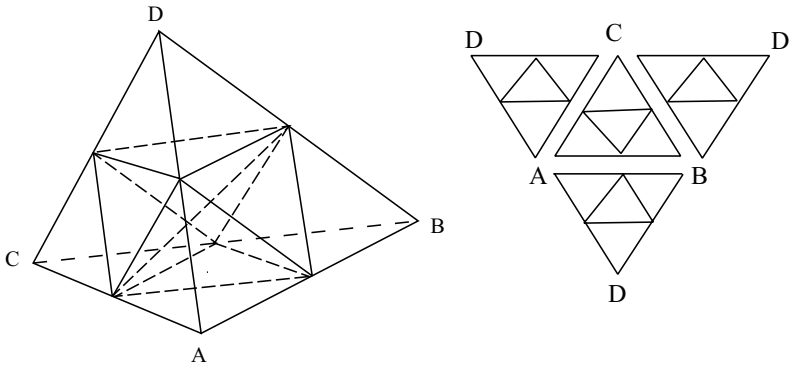


Figura 2. Partición semejante en 3D

Definición 8 *(Partición semejante) El tetraedro original es dividido en ocho subtetraedros cortando las cuatro esquinas por planos paralelos a la cara opuesta y dividiendo el octaedro interior resultante en cuatro tetraedros más (Figura 2).*

Nótese que en la división del octaedro es importante la elección de la arista interna de manera que se optimice la forma de los cuatro tetraedros interiores[4]. De los 8 tetraedros resultantes sólo los 4 de las esquinas son semejantes al original.

Definición 9 *(Partición 8T-LE) Para un tetraedro t con arista mayor única, la partición 8T-LE (8 Tetraedra-Longest Edge partition) de t se divide en 8 tetraedros mediante: bisección por la arista mayor de t produciendo dos tetraedros t_1, t_2, bisección de cada t_i por el punto medio de la única arista mayor de t_i que es arista mayor de la cara común de t_i con el tetraedro original t y por último bisección de cada uno de los 4 tetraedros generados en el paso anterior por el punto medio de la única arista común con una arista del tetraedro original.*

La Figura 3 muestra uno de los posibles casos al aplicar la partición 8T-LE. En el Apéndice se dan todas las configuraciones posibles producidas por esta partición.

El esqueleto de una triangulación y los patrones de división de n-símplices nos permiten presentar un esquema general para algoritmos de refinamiento basado en el esqueleto. Aunque nuestra versión de estos algoritmos está basada en las particiones 4T-LE para el

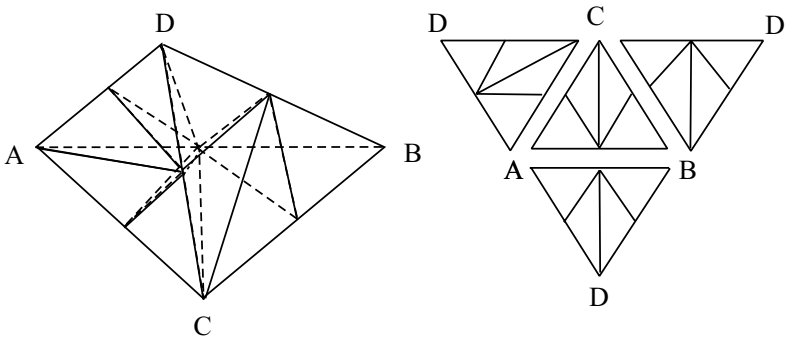


Figura 3. Ejemplo de partición 8T-LE

caso bidimensional y 8T-LE para el tridimensional, el esquema general siguiente no precisa de hipótesis concretas sobre la partición utilizada y podría ser utilizado con cualquiera de las presentadas con anterioridad. La idea procede de considerar los sucesivos esqueletos en orden inverso. De ahí que el procedimiento básico para un símplice n-dimensional consiste en los n pasos siguientes:

Refinamiento basado en el esqueleto (malla, n-simplice, particion)
Para cada n-simplice que ha de ser refinado **hacer**
1) Division del 1-esqueleto
2) Division del 2-esqueleto
. . .
n) Reconstruccion del n-simplice
Fin Para

Se debe notar que el último paso del algoritmo anterior sobre la reconstrucción o subdivisión adecuada del n-símplice necesita alguna información geométrica (tal como puntos interiores, aristas o caras y las conexiones entre ellas) para llevar a cabo la subdivisión correctamente. Esta información es específica a la partición particular usada y se supone que está incluida en el parámetro de entrada general de nombre *partición*. En la siguiente sección presentamos el grafo del esqueleto.

EL GRAFO BASADO EN EL ESQUELETO DE UNA TRIANGULACIÓN

Partiendo del concepto de esqueleto de un símplice es posible construir un grafo que representa las adyacencias entre las k-caras de una triangulación. En la Figura 4 se muestra una representación jerárquica de los distintos esqueletos de una triangulación. En la Figura 4a se tiene el tetraedro de referencia en R^3. La Figura 4b muestra las caras una vez abierto el tetraedro a lo largo de las tres aristas que comparten el vértice D y proyectadas en el plano. Como consecuencia resulta una representación del 2-esqueleto del tetraedro. En la Figura 4c se representan las aristas del tetraedro original formando el 1-esqueleto. Finalmente la Figura 4d muestra los cuatro vértices que definen el 0-esqueleto del tetraedro.

Definición 10 *(Grafo del k-esqueleto $G^k(P,L)$) Sea τ una malla n-simplicial con sus correspondientes conjuntos de k-esqueletos con $k < n$. El grafo no-dirigido del k-esqueleto $G^k(P,L)$ de τ se construye como sigue. El conjunto de los nodos del grafo $P = \{p_0, p_1, ..., p_r\}$ es el conjunto de las k-caras de τ. Sea $L = \{l_0, l_1, ..., l_s\}$ tal que $l_j \in L$ es el enlace en el grafo que representa una relación dada R entre dos nodos diferentes p_n, p_m de P. Denotamos $l_i \sim (p_m, p_n)$.*

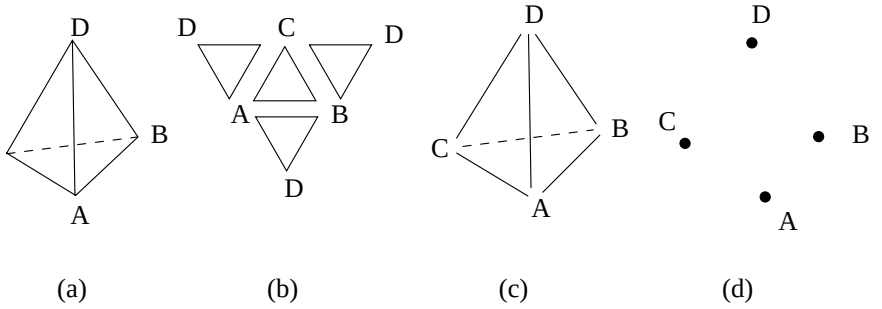


Figura 4. Representación gráfica de los k-esqueleto

Sea R la relación de adyacencia topológica entre dos k-caras de τ. Es decir, para dos nodos diferentes p_n, p_m de P, p_n R p_m, si y sólo si la k-cara representada por p_n comparte con p_m una *(k-1)*-cara común. Por brevedad denotamos por $G^k(\tau)$ el grafo del k-esqueleto. Las Figuras 5a.1 y 5b.1 muestran los grafos no dirigidos del 1- y del 2-esqueleto para un triángulo y un tetraedro.

A los enlaces del grafo se les pueden añadir etiquetas. Para cada enlace $l_i \in L$ y nodos p_n,$p_m \in P$, tal que $l_i \sim (p_m,\ p_n)$ se asigna un entero que identifica la $(k+1)$-cara o $(k+1)$-símplice, en el cual los miembros del k-esqueleto representados por $(p_m,\ p_n)$ están contenidos. De esta forma se puede reconocer la relación "hacia arriba" de los elementos del k-esqueleto como por ejemplo arista $\rightarrow$ cara, cara $\rightarrow$ tetraedro. Esto es muy útil al programar los algoritmos de refinamiento. Además, si se precisa mantener una relación como por ejemplo arista $\rightarrow$ tetraedro, la etiqueta será un único número entero que refiere al tetraedro adecuado de la malla. Esta característica es apropiada para las estructuras de datos de los algoritmos de refinamiento/desrefinamiento, porque mantiene enlaces 'hacia arriba' en la representación jerárquica del esqueleto.

Consideremos ahora como ejemplos el triángulo y el tetraedro y sus grafos basados en los 1- y 2-esqueletos. La especificación de los enlaces de los nodos en el grafo está determinada por la relación de adyacencia R. En la Figura 5 se muestran las representaciones de los grafos basados no sólo en la relación R sino en todas las relaciones posibles para las conexiones entre aristas y caras en 2D y 3D respectivamente. Especificaciones diferentes del grafo dependen de las relaciones R entre las k-caras.

Para manipular las configuraciones de la Figura 5 se utiliza un vector de enteros T^{n-1}, $(n = 2, 3)$, que tiene por elementos los diferentes grados de los nodos del grafo del $(n-1)$-esqueleto. La longitud de T^{n-1} es el número de elementos en el $(n-1)$-esqueleto (3 para un triángulo y 4 para un tetraedro). El orden en T^{n-1} no es relevante, así que cualquier permutación de T^{n-1} es válida. Por ejemplo, para los grafos de las Figuras 5a.1 y 5b.4 se pueden usar los vectores $T^1 = [2, 2, 2]$ y $T^2 = [2, 2, 3, 1]$ respectivamente. La importancia de estos vectores es que determinan R y por tanto especifican las relaciones de adyacencia topológica en la malla. Por ejemplo, $T^1 = [2, 2, 2]$ y $T^2 = [3, 3, 3, 3]$ implican que se mantendrán todas las relaciones topológicas posibles entre las k-caras de la malla (cada arista con orden dos en un triángulo y cada cara con orden tres en el tetraedro). Sin embargo, la adyacencia definida por T^{n-1} puede ser restringida a alcanzar una computación más eficiente mediante, por ejemplo minimizando el número de relaciones que hay que almacenar para un único n-símplice. Así que el siguiente funcional es útil para este propósito

$$T_n = \min_i \sum_j T_i^{n-1}(j)$$

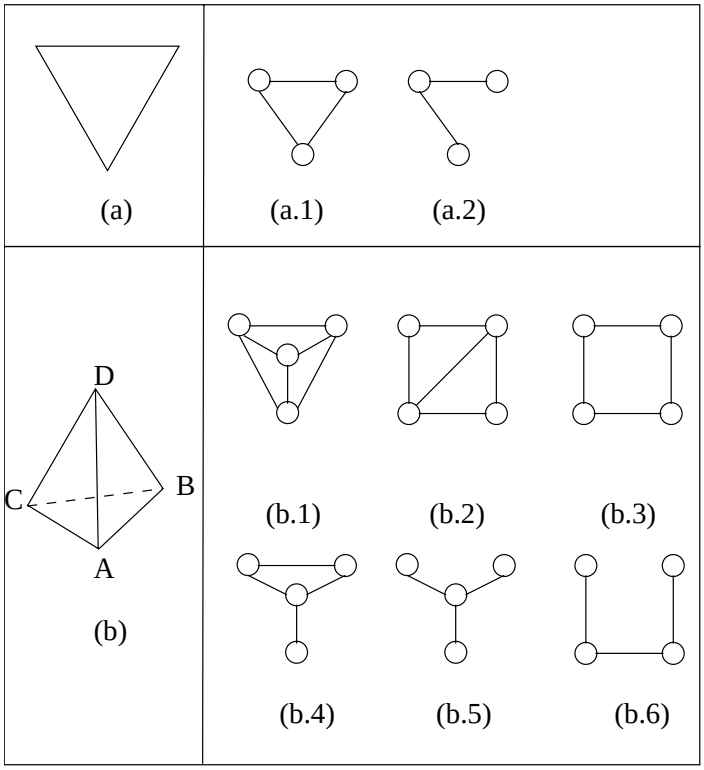


Figura 5. Configuraciones posibles para el grafo del $(n-1)$-esqueleto

donde el índice i representa los diferentes vectores T^{n-1} para una dimensión dada n (n=2,3) y j indexa cada elemento en el vector. T_n minimiza el número de enlaces en el grafo del esqueleto para el símplice. Las configuraciones de G^{n-1} para T_n se muestran en la Figura 6.

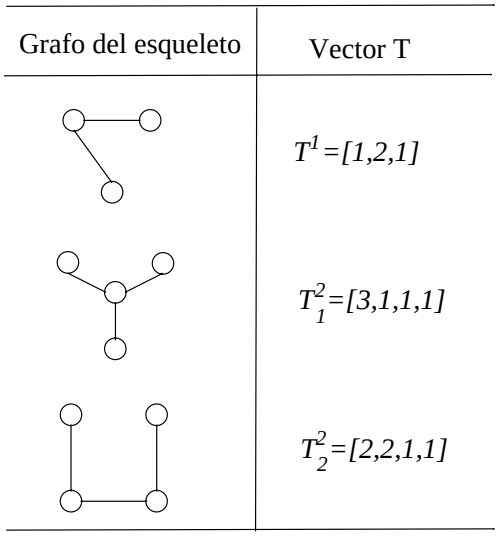
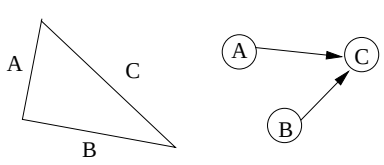


Figura 6. Grafo del esqueleto y vector T

Para nuestro objetivo, uso del grafo en la modelización del esquema de refinamiento basado en la bisección por la arista mayor, restringiremos la relación R entre las k-caras del grafo del (n-*1*)-esqueleto a aquellas en la Figura 6. En la próxima sección justificamos esta elección.

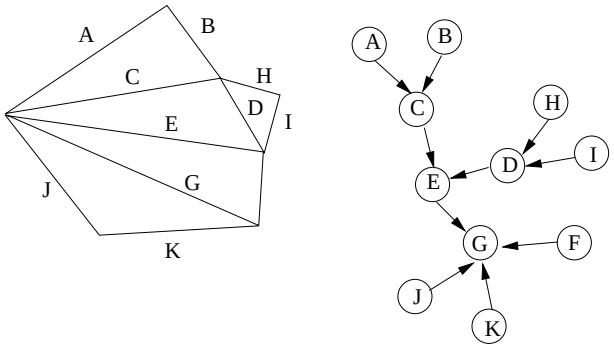
El Grafo del 1-esqueleto para los algoritmos de bisección por la arista mayor en 2D

Los algoritmos de refinamiento basados en la bisección por la arista mayor de un triángulo en 2D tienen la ventaja de que las diferentes posibilidades de refinar un triángulo dependen de determinar la arista mayor. En general, se puede decir que estos algoritmos clasifican las aristas de cada triángulo en dos tipos: la arista mayor, de *tipo 1*, y las otras dos aristas, de *tipo 2*, y esto es suficiente para nuestros propósitos (Figura 7a).



Arista	Tipo
A	2
B	2
C	1

(a)



(b)

Figura 7. Grafo del 1-esqueleto en 2D

Esta idea se puede expresar en términos del grafo del 1-esqueleto presentado antes al considerar la relación R, de forma que xRy, si y sólo si y es la arista mayor del triángulo con arista x y además considerando el vector T^1 en la Figura 6. Como R es una relación de orden entre las aristas, es posible añadir una orientación a los enlaces del grafo que representan este orden. Por tanto, los enlaces en la Figura 7a indican la dependencia de las aristas al refinar y para asegurar la conformidad. En la Figura 7b se muestran una sencilla triangulación y el grafo asociado basado en las aristas.

Proposición 1 *Sea* G^1 *el grafo dirigido del 1-esqueleto de una triangulación bidimensional conforme* τ*, entonces el grafo no contiene ciclos.* □

Demostración: Hay dos situaciones en las que pueden ocurrir bucles (ciclos). La primera aparece en mallas que tienen triángulos isósceles o equiláteros. Al momento de construir

el grafo, en los casos en los que hay triángulos isósceles o equiláteros, se puede dar más de una arista de bisección (arista mayor). En estas situaciones, en las que se puede elegir más de una arista mayor, la primera asignada como arista mayor es la elegida para que sea la arista mayor. La segunda situación está descrita en la Figura 8. Sea X=$\{x_0, x_1, x_2, x_3, \cdots, x_n\}$ las aristas comunes compartidas por triángulos adyacentes en el *LEPP* de t_0 (t_0 es el triángulo definido por aristas c y x_0). Si aparece un ciclo, entonces $long(c) \leq long(x_0) \leq \ldots \leq long(x_n) \leq long(c)$. Pero esto es imposible si las aristas de X son de distinta longitud. Si las aristas de X son de igual longitud, estamos en el primer caso considerado. □

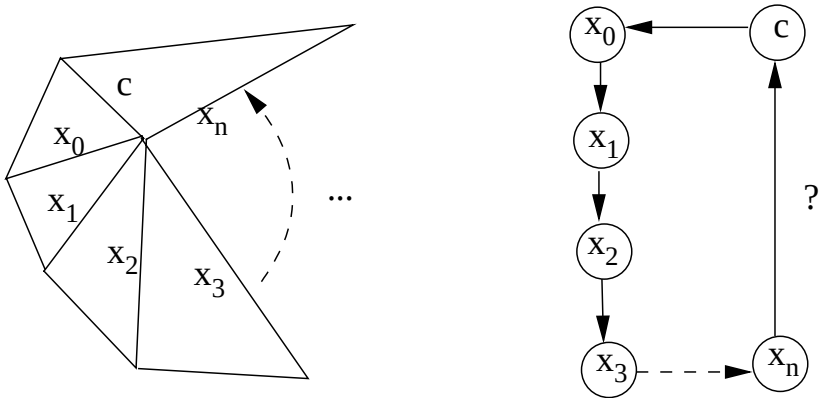


Figura 8. Existencia de bucles en el grafo del del 1-esqueleto

Desde el punto de vista computacional la siguiente Proposición asegura la localidad de la estructura de datos basada en el grafo dirigido del esqueleto[25].

Proposición 2 *El coste de actualización de la estructura de datos basada en el grafo dirigido del 1-esqueleto al aplicar la bisección 4T-LE es como mucho tres nodos en cada paso de refinamiento.* □

Proposición 3 *Sea τ una triangulación conforme bidimensional con N triángulos y t un triángulo arbitrario de τ. Sea e la arista mayor de t. Entonces el LEPP de t se puede obtener de un recorrido en profundidad del grafo dirigido del 1-esqueleto de τ, comenzando por el nodo e del grafo con un coste máximo del orden de O(N-1).* □

El grafo del 1- y 2-esqueleto para los algoritmos de bisección por la arista mayor en 3D

Consideraremos primero el grafo del 1-esqueleto de un tetraedro. Como en 2D, se aplica el mismo criterio en cuanto a la relación entre las aristas. En este caso se pueden distinguir tres tipos de aristas de acuerdo con su longitud (esta clasificación es semejante a la de Bänsch[2], Liu y Joe[17] y Mukherjee[21]). La arista mayor del tetraedro la llamamos arista *tipo 1*. La arista mayor de cada una de las dos caras que no comparten la arista mayor del tetraedro son arista *tipo 2* y el resto de aristas son aristas *tipo 3*[23].

Observación 1 *Mediante pequeñas (e hipotéticas) perturbaciones de las coordenadas de los nodos puede asegurarse que hay exactamente una única arista mayor en cada tetraedro. La arista mayor de un tetraedro es frecuentemente llamada arista de referencia.*

Para la partición 8T-LE, distinguimos tres tipos de tetraedros dependiendo de la posición relativa de sus tipos de aristas [23]. Primero consideramos los tetraedros obtenidos al

bisectar la arista tipo 1, después la o las aristas tipo 2 y así sucesivamente. Cada tipo de tetraedro se puede representar por un grafo orientado del 1-esqueleto como en la Figura 9. En el Apéndice se muestran todas las posibles configuraciones para la partición 8T-LE y los grafos orientados del 1-esqueleto asociados.

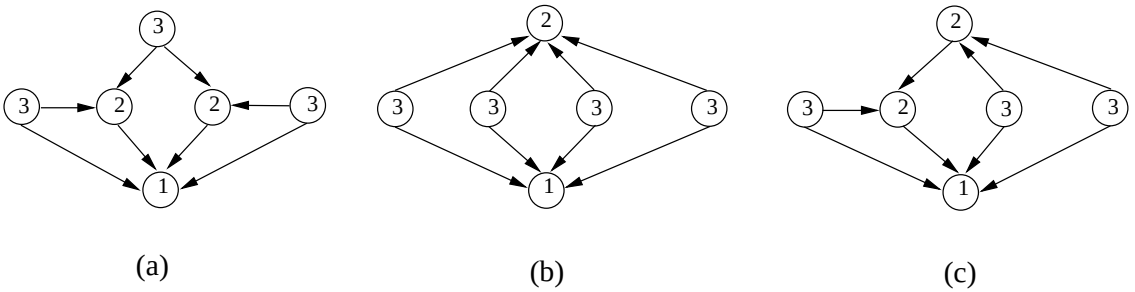


Figura 9. Grafo orientado G^1 para los tres tipos de tetraedros

Observación 2 *Algunas propiedades referentes al grafo dirigido del 1-esqueleto para un tetraedro son:*

1. *El grafo dirigido del 1-esqueleto tiene exactamente 6 nodos y 8 enlaces.*
2. *El grado del nodo que representa la arista mayor es 4.*
3. *El grafo dirigido del 1-esqueleto es planar y desconectado y no contiene ciclos.*
4. *Cada camino tiene longitud 1.*

El grafo dirigido del 1-esqueleto de un tetraedro como ha sido presentado anteriormente es una representación basada en las aristas y se puede utilizar para diseñar una estructura de datos de cualquier esquema de refinamiento de tetraedros basado en bisección por la arista mayor. Esto nos proporciona la información necesaria para llevar a cabo el refinamiento de un tetraedro en término de sus aristas.

Desde el punto de vista computacional se puede diseñar el algoritmo de refinamiento mediante la construcción local del grafo de forma dinámica. Esta es la idea que usamos aquí por eficiencia en la memoria utilizada. De otra forma, se podría construir el grafo para cada tetraedro en la malla y entonces actualizar el grafo tras cada refinamiento. Esto implica un almacenamiento extra de datos de *8n,* siendo n el número de tetraedros en la malla.

En la referencia 22 se estudian resultados asintóticos sobre las relaciones de adyacencia entre los elementos topológicos de las mallas en 3D al aplicar la partición 8T-LE. Por ejemplo, cada arista es compartida por 36/7 tetraedros en término medio. Teniendo esto en cuenta, en el grafo del 1-esqueleto un nodo podría tener en media hasta $\left(\frac{36}{7}\right) \times 4$ enlaces. Esto implicaría unas necesidades de almacenamiento inaceptables, y por tanto no es eficiente mantener el grafo del 1-esqueleto grafo globalmente, sino localmente para cada tetraedro. Ahora estudiaremos el grafo del 2-esqueleto como una estructura de datos alternativa para ser extendida globalmente a toda la malla.

Observación 3 *La representación de las Figuras 10a-b es única. En este caso la arista AB es la arista mayor. Esta representación se llama posición estándar*[2]*. Llamamos a las caras f1 y f4 de la figura caras de referencia.*

Tenemos dos alternativas para construir el grafo del 2-esqueleto para un único tetraedro (Figura 10c-d). Usando la notación del vector presentado al principio de la Sección, el

grafo de la Figura 10c se escribe como $T_1^2 = [3, 1, 1, 1]$. En este caso el nodo con grado 3 es una de las dos caras de referencia (si AB es la arista mayor en el tetraedro, entonces las caras de referencia son *f1* y *f4*). La segunda posibilidad se muestra en la Figura 10d y es denotada por $T_2^2 = [2, 2, 1, 1]$. Ahora cada nodo con grado 2 representa una cara de referencia (compartiendo la arista de referencia del tetraedro).

Para construir el grafo no dirigido del 2-esqueleto $G^2(\tau)$ para una triangulación τ en 3D se pueden tomar dos posibles configuraciones para las caras de cada tetraedro. Ambas configuraciones son idénticas para nuestros objetivos y tienen las mismas propiedades respecto al grafo final. Mediante la construcción de dicho grafo $G^2(\tau)$ se puede deducir una estructura de datos basada en las caras que es adecuada a las necesidades del algoritmo 3D-SBR, como explicará la siguiente sección.

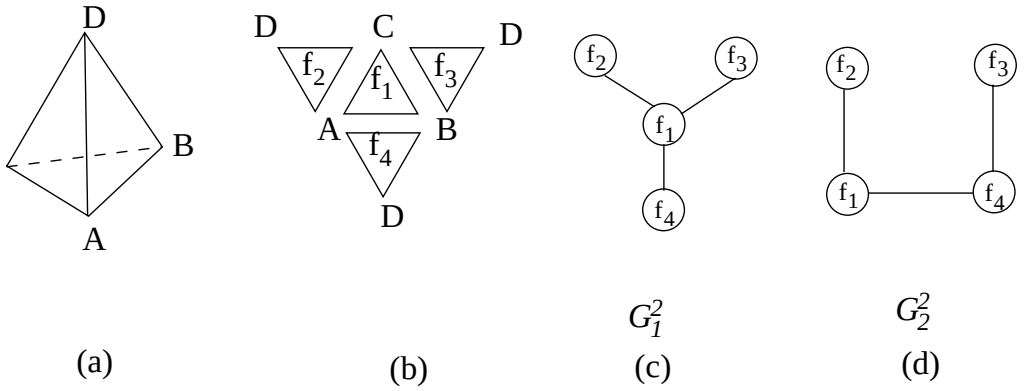


Figura 10. 2-esqueleto para un tetraedro

Proposición 4 *Sea $G^2(\tau)$ el grafo del 2-esqueleto no dirigido de una triangulación tridimensional τ, entonces el grafo es planar y conectado.* □

LOS ALGORITMOS BASADOS EN EL ESQUELETO EN 2D Y 3D

En esta sección se presentan los algoritmos de refinamiento basados en el esqueleto en 2D y 3D teniendo en cuenta la estructura de datos tipo grafo introducida previamente. Se ha incluido el algoritmo 2D en un paquete de ecuaciones en derivadas parciales de MatLab, PDE Toolbox. Este paquete usa un esquema para el refinamiento que es una combinación de la partición semejante y de la partición simple por el lado mayor.

Las particiones usadas al refinar la triangulación en nuestros algoritmos son la 4T-LE para el caso 2D y la 8T-LE para 3D. Con el grafo basado en el esqueleto los algoritmos admiten una descripción más natural y consistente.

El algoritmo 2D-SBR[23] básicamente realiza el refinamiento mediante dos pasos secuenciales: identifica y bisecta las aristas especificadas por los indicadores de refinamiento (ver pasos 1 y 2, 2D-SBR más abajo) y subdivide los triángulos individuales para definir una nueva triangulación, (paso 3 abajo). El procedimiento *subdivisión* subdivide o bien una arista o bien un triángulo. La subdivisión de una arista consiste en la bisección de la arista por el punto medio. La subdivisión de un triángulo depende del número de aristas bisectadas. Si tiene sólo una arista dividida, entonces el triángulo es dividido por la arista mayor. Si tiene dos aristas divididas, primero se realiza la división por la arista mayor y después se unen los puntos medios de las dos aristas divididas. Por último, si tiene las tres aristas divididas, se aplica la partición 4T-LE.

Algoritmo 2D-SBR (τ, t_0)
/* Entrada: malla τ y triangulo a refinar t_0
/* Salida: malla τ
1: $L = 1 - esqueleto(t_0)$
Para cada arista $e_i \in L$ **hacer**
Subdivision(e_i)
Fin Para
2: Para cada arista $e_i \in L$ **hacer**
$S_i = LEPP(e_i, G^1(\tau))$
Para cada triangulo $t_i \in S_i$ **hacer**
Sea e_i la arista mayor de t_i
Subdivision(e_i)
Fin para
3: Para cada triangulo $t_j \in \tau$ a subdividir **hacer**
Subsivision(t_j)
Fin para
Fin algoritmo

El algoritmo 2D-SBR tiene complejidad lineal con respecto al número de nodos. Tiene dos puntos costosos desde el punto de vista computacional. El primero proviene de la creación y actualización de datos y su coste depende de la malla inicial. Las actualizaciones de datos subsiguientes son locales a cada triángulo. El segundo punto corresponde al paso 2 para la identificación y bisección de las aristas que intervienen en el proceso de refinamiento. El caso peor ocurre si la triangulación inicial es tal, que cada triángulo en el camino de propagación tiene su arista mayor mayor que el triángulo anterior. Los experimentos numéricos muestran que este coste se acerca de forma asintótica a una pequeña constante a medida que la malla es refinada.

Algoritmo 3D-SBR (τ, t_0)
/* Entrada: malla τ y triangulo a refinar t_0
/* Salida: malla τ
1: $L = 1 - esqueleto\ (t_0)$
Para cada arista $e_i \in L$ **hacer**
Subdivision (e_i)
Fin Para
2: Mientras $L \neq \emptyset$ **hacer**
Sea $e_j \in L$
Para cada tetraedro $t_i \in hull(e_j)$ **hacer**
Para cada cara no-conforme $f_i \in G^2(t_i)$ **hacer**
Sea e_p la arista mayor de f_i
Subdivision (e_p)
$L = L \cup e_p$
Fin para
Fin para
$L = L - e_j$
Fin mientras
3: Para cada cara $f_i \in G^2(\tau)$ a subdividir **hacer**
Subsivision (f_i)
Fin para
4: Para cada tetraedro $t_i \in \tau$ a subdividir **hacer**
Subsivision (t_i)
Fin para
Fin algoritmo

Para el algoritmo 3D-SBR las cuatro etapas principales son: (1) subdivisión de las aristas del tetraedro de entrada; (2) subdivisión de las aristas debido a la propagación

por conformidad; (3) subdivisión de las caras triangulares por la partición 4T-LE; (4) subdivisión del interior de los tetraedros que han tomado parte en el refinamiento mediante la partición 8T-LE y las particiones de refinamiento local asociadas.

El procedimiento *subdivisión* subdivide los sucesivos esqueletos en orden *inverso*: los pasos 1 y 2 llevan a cabo la subdivisión de las aristas, el paso 3 realiza la subdivisión de las caras y el paso 4 subdivide el interior de los tetraedros.

También en 3D el algoritmo es de complejidad lineal en el número de nodos[23]. Los puntos donde se usan el grafo del 2-esqueleto $G^2(\tau)$ se ven claramente en el esquema anterior. Primero, el bucle interior del paso 2 accede a las caras no conformes de la malla y en el paso 3 en la subdivisión del 2-esqueleto. En ambos puntos el grafo del 2-esqueleto proporciona acceso a las caras que toman parte en el refinamiento de forma local y eficiente.

En el paso 2 el procedimiento *hull(e)* proporciona el conjunto de símplices $S \in \tau$ tal que $e \in S$. Es decir, *hull(e)* es el conjunto de los símplices que comparten la misma arista e. En la referencia 23 se da un algoritmo para encontrar la envoltura de una arista (*hull*) de complejidad lineal.

Las versiones anteriores de los algoritmos 2D/3D-SBR realizan el refinamiento de un solo elemento. Obviamente, los algoritmos se pueden usar para el refinamiento de un conjunto de elementos, simplemente usando ese conjunto de elementos como entrada en los algoritmos en lugar de un único elemento.

EXPERIMENTOS CON EL ALGORITMO BASADO EN EL ESQUELETO. CASO 2D

En la referencia 24 se presenta el algoritmo inverso del 3D-SBR, el 3D-SBD, y en la referencia 25 se describe la aplicación del algoritmo 3D-SBR combinado con el algoritmo de desrefinamiento.

A continuación se presentan y discuten algunos resultados numéricos en 2D. Consideramos el dominio que representa la isla de Gran Canaria y refinamiento local usando la partición 4T-LE en tres subdominios adyacentes. Se ha implementado la estructura de datos basada en el grafo. El ejemplo nos proporciona: (1) una demostración del algoritmo basado en el esqueleto tal como ha sido descrito en las secciones anteriores; (2) un estudio del tiempo utilizado por el algoritmo que para este problema de test muestra la variación en el tiempo total de CPU, cuando el número de nodos crece con cada refinamiento de malla, y (3) algunas medidas estadísticas del subgrafo del *LEPP* determinado por el grafo en cada paso de refinamiento.

En este primer ejemplo consideramos el dominio de la isla de Gran Canaria y tres subdominios S1, S2 y S3, que corresponden a zonas en las que la elevación del terreno es cada vez mayor.

Se han realizado seis niveles de refinamiento: partiendo de una malla inicial (nivel 1, 592 elementos) de la Figura 11, que es una malla tipo Delaunay, se refinan todos los elementos del subdominio más interior S3 y se obtiene la malla del nivel 2. Obsérvese que el refinamiento se propaga fuera del subdominio S3 con el fin de asegurar la conformidad de la triangulación. El nivel 2 de malla contiene 736 elementos. En el segundo paso se eligen de nuevo los elementos del subdominio S3 para ser refinados con propagación fuera de S3 como antes. Se obtiene la malla nivel 3 con 1230 elementos. En el tercer paso se eligen los elementos del subdominio S2 y se obtiene la malla del nivel 4 con 2624 elementos. En el cuarto paso elegimos todos los elementos de S1$\cup$S2 y obtenemos la malla de nivel 5 con 9258 triángulos. Ahora se eligen todos los elementos de S2$\cup$S3 y se obtiene la malla de nivel 6 con 30730 elementos. Por último se refinan los elementos del subdominio S1 y obtenemos la malla del nivel 7 con 41448 elementos.

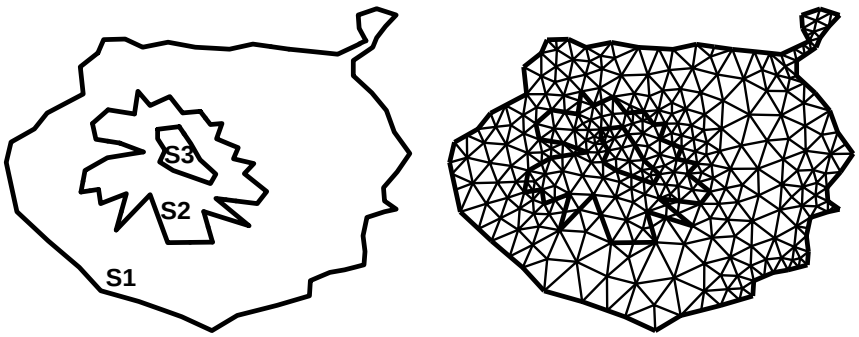


Figura 11. Isla de Gran Canaria mostrando los tres subdominios S3, S2, S1 y malla asociada tipo Delaunay

Se ha utilizado el algoritmo 2D-SBR y la estructura de datos del grafo, tal como han sido descritos anteriormente. Estos resultados muestran efectivamente el uso de la estructura de datos basada en el grafo.

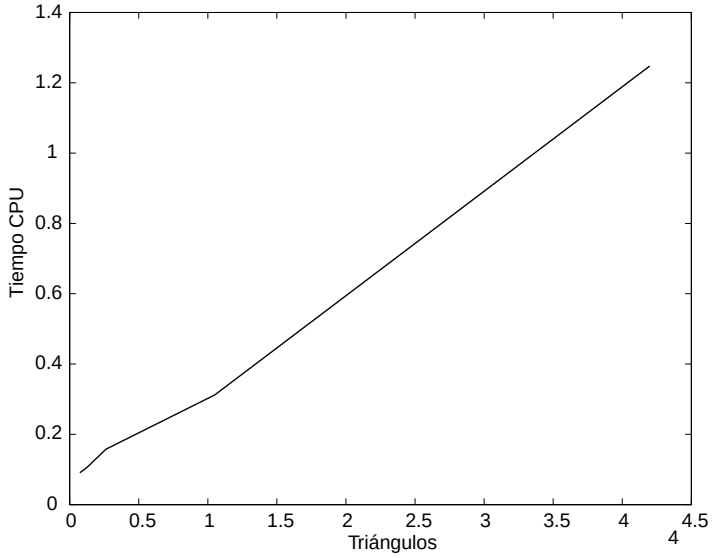
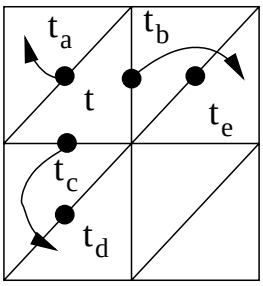


Figura 12. Tiempo de CPU para el problema de test

A continuación estudiamos el tiempo de CPU con relación al número de nodos para cada malla obtenida antes. Se tiene una complejidad lineal tanto para el algoritmo de refinamiento como para la estructura de datos basada en el grafo. Estas se muestran en la Figura 12.

Finalmente, se dan algunas estadísticas sobre el comportamiento del *LEPP* para este algoritmo. El *LEPP* se calcula directamente del subgrafo de la estructura de datos basada en el grafo. Aquí se ofrecen resultados usando dos métricas. La primera es el número adicional de triángulos refinados al refinar t. La segunda métrica es la longitud máxima de los *LEPP's* de los vecinos de t sin contar el triángulo t. Nos referiremos a estas métricas como M1 y M2 respectivamente. Por ejemplo, M1 = 5 y M2 = 2 al refinar el triángulo t de la Figura 13 y además son valores típicos en la práctica.



Triángulo	LEPP
t	$\{t,t_a\}$
t_a	$\{t_a,t\}$
t_b	$\{t_b,t_e\}$
t_c	$\{t_c,t_d\}$

Figura 13. *LEPP* al refinar un triángulo

En las Tablas I y II se presentan la media, desviación típica y valores máximos para M1 y M2. Los resultados de la Tabla I muestran que M1, el número de elementos adicionales que se recorren al refinar un único elemento, tiende en media a 5. Esto significa que el grafo basado en el 1-esqueleto presentado en la página 98 contiene en media aproximadamente 5 nodos por elemento. La columna Max M1 proporciona el valor máximo de M1 para los elementos de la malla en cada nivel de refinamiento. Este número se reduce en un 50 % desde la malla inicial (nivel 1) a la última malla (nivel 7). Estos resultados así como los experimentos anteriores muestran que el coste del refinamiento local para el algoritmo 2D-SBR decrece, cuando el número de elementos refinados crece.

Nivel	N. Elem.	Media M1	Std. M1	Max. M1
1	592	6,6199	2,7647	20
2	736	6,6902	2,5771	20
3	1230	6,5057	2,2658	20
4	2624	6,0191	1,6793	15
5	9258	5,5134	1,1629	11
6	30730	5,2478	0,6814	11
7	41448	5,2128	0,5642	10

Tabla I. Estadísticas de la métrica M1 para la malla de Gran Canaria

Nivel	N. Elem.	Mean M2	Std. M2	Max. M2
1	592	3,4510	1,6600	10
2	736	3,5394	1,6631	10
3	1230	3,3837	1,6017	9
4	2624	2,9184	1,0519	8
5	9258	2,4265	0,6017	7
6	30730	2,3672	0,5090	7
7	41448	2,1891	0,3287	7

Tabla II. Estadísticas de la métrica M2 para la malla de Gran Canaria

Los resultados de la Tabla II son relativos a la métrica M2. El valor medio de la longitud del camino de propagación por la arista mayor tiende a 2 y el máximo de estos números decrece un 30 % desde la malla inicial hasta la final. Esto implica que, en términos del grafo del 1-esqueleto, el coste para obtener el *LEPP* al refinar un solo elemento tiende a 2 en media.

El uso de estas métricas para el *LEPP* tiene mucho interés, ya que la utilidad de los esquemas de bisección por la arista mayor y el uso eficiente de memoria puede ser cuestionado. La justificación teórica de estos resultados forma parte de un trabajo que se está llevando a cabo actualmente.

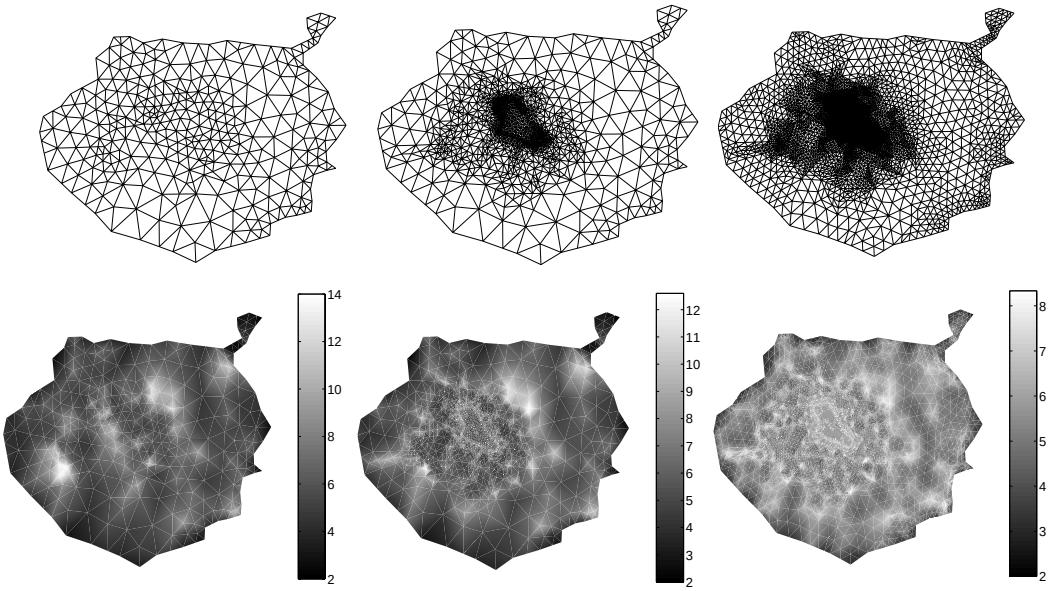


Figura 14. Mallas de Gran Canaria correspondientes a niveles de refinamiento 1, 4, 5 y distribución espacial de M1

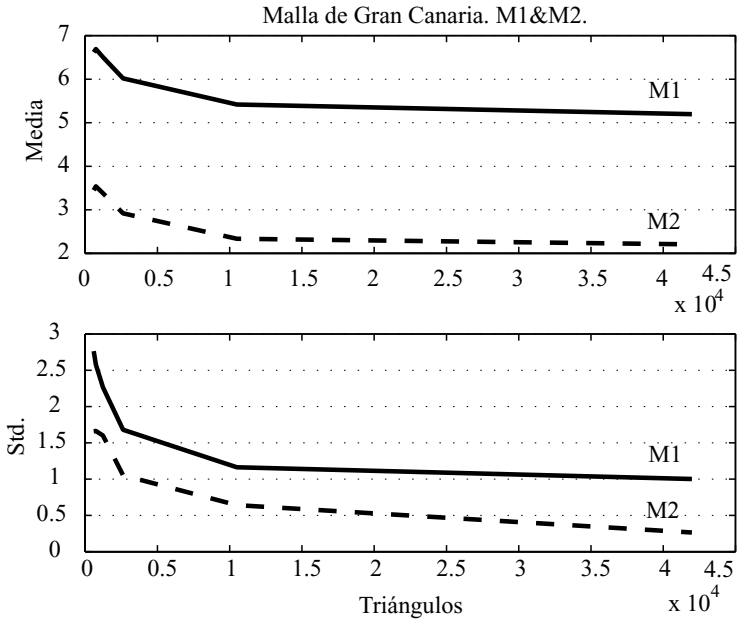


Figura 15. Estadísticas de la Malla de Gran Canaria

CONCLUSIONES

En este artículo se han revisado algunas ideas para el refinamiento en dos y tres dimensiones, extendiendo nuestro trabajo previo en los algoritmos de refinamiento basados en el esqueleto. Para el algoritmo 2D-SBR se ha presentado una estructura de datos basada en grafos. Algunas propiedades de esta estructura de datos se han descrito mostrando su eficiencia en cuanto almacenamiento y tiempo de coste. También se ha presentado la extensión de la estructura de datos basada en el grafo al caso de dimensión tres así como una estructura de datos basada en las caras.

Se han discutido con detalle los grafos para los refinamientos de triangulaciones en 2D y 3D. Finalmente, se han mostrado experimentos numéricos para ilustrar las ideas presentadas usando el algoritmo 2D-SBR.

REFERENCIAS

1 R.E. Bank y A.H. Sherman, "The use of adaptive mesh refinement for badly behaved elliptic differential equations", in R. Vichnevetsky y R.S. Stepleman (Eds.), *Advances in computer methods for partial differential equations III*, pp. 33–39, (1979).

2 E. Bänsch, "Local mesh refinement in 2 and 3 dimensions", *IMPACT Com. Sci. Engng.*, Vol. **3**, pp. 181–191, (1991).

3 E.B. Becker, Graham F. Carey y J. Tinsley Oden, "*Finite elements*", Prentice-Hall, (1981).

4 J. Bey, "Simplicial mesh refinement: on Freudenthal's algorithm and the optimal number of congruence classes", *Numer. Math.*, (2000). URL:http://dx.doi.org/10.1007/5002111111108.

5 J. Bonet y J. Peraire, "An alternating digital tree (ADT) algorithm for 3D geometric and intersection problems", *Int. J. Num. Meth. Engng.*, Vol. **31**, pp. 1–17, (1991).

6 G.F. Carey, "Computational grids: generation, adaptation and solution strategies", Taylor & Frances, (1997).

7 G.F. Carey y A. Pehlivanov, "An adaptive octree-based scheme for hierarchic extraction compression and remote visualization of data", *Fifth USNCCM, Boulder*, CO, 4-6 Agosto (1999).

8 G.F. Carey, A.I. Pehlivanov, R.C. Mastroleo, R. McLay, Y. Shen, V. Carey y R. Dutton, "Hierarchic visualization and parallel computation", *Proceedings High performance computing'98*, Boston, MA, Abril (1998).

9 G.F. Carey, M. Sharma y K.C. Wang, "A class of de data structures for 2-D and 3-D adaptive mesh refinement", *Int. J. Num. Meth. Engng.*, Vol. **26**, pp. 2607–2622, (1988).

10 G.F. Carey, "A mesh refinement scheme for Finite Element computations", *J. Comput. Meth. Appl. Mech. Engng.*, Vol. **7**, N^o 1, pp. 93–105, (1976).

11 J.L. Gross y T.W. Tucker, "Topological graph theory", John Wiley & Sons, (1987).

12 F. Haray, "*Graph theory*", Addison Wesley, (1972).

13 Y. Kallinderes y P. Vijayan, "Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes", *AIAA J.*, Vol. **31**, pp. 1440–1447, (1993).

14 L. Kettner, "Using generic programming for designing a data structure for polyhedral surfaces", *Comp. Geom.- Theory and Applications*, Vol. **13**, pp. 65–90, (1999).

15 I. Kossaczký, "A Recursive approach to local mesh refinement in two and three dimensions", *J. Comp. Appl. Math.*, Vol. **55**, pp. 275–288, (1994).

16 P. Leinen, "Data structures and concepts for adaptive finite element methods", *Computing*, Vol. **55**, pp. 325–354, (1995).

17 A. Liu y B. Joe, "Quality local refinement of tetrahedral meshes based on bisection", *SIAM J. Sci. Statest. Comput.*, Vol. **16**, pp. 1269–1291, (1995).

18 R. Löhner, "Some useful data structures for the generation of unstructured grids", *Comm. Appl. Num. Meth.*, Vol. **4**, pp. 123–135, (1988).

19 R. Löhner, "Edges, star, super-edges and chains", *Comput. Meth. Appl. Mech. Engng.*, Vol. **11**, pp. 255–263, (1994).

20 E. Muthukrishnan, P.S. Shiakolas, R.V. Nambiar y K.L. Lawrence, "Simple algorithm for adaptive refinement of three-dimensional finite element tetrahedral meshes", *AIAA Journal*, Vol. **33**, pp. 928–932, (1995).

21 A. Mukherjee, "An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity", Tesis doctoral, Penn. State University, (1996).

22 A. Plaza, "Asymptotic behavior of the average of the adjacencies of the topological entities in some simplex partitions", in *8th International Meshing Roundtable*, South Lake Tahoe, California, 10-13 Octubre (1999).

23 A. Plaza y G.F. Carey, "Local refinement of simplicial meshes based on the skeleton", *Appl. Num. Math.*, Vol. **32**, pp. 195–218, (2000).

24 A. Plaza y M.A. Padrón, "Un algoritmo de desrefinamiento en 3D para mallas de tetraedros basado en el esqueleto", *Rev. Int. Mét. Num. Cálc. Dis. Ing.*, Vol. **15**, N^{o} 4, pp. 471–483, (1999).

25 A. Plaza, M.A. Padrón y G.F. Carey, "A 3D refinement/derefinement algorithm for solving evolution problems", *Appl. Num. Math.*, Vol **32**, pp. 401–418, (2000).

26 A. Plaza, J.P. Suárez y M.A. Padrón, "Mesh graph structure for longest-edge refinement algorithms", *Sandia Report SAND 98-2250*, pp. 335–344, (1997).

27 M.C. Rivara y A. Plaza, "Mesh selective refinement/derefinement based on the 8-tetraedros longest-edge partition", Dept. Ciencia Computacional, Universidad de Chile, TR/DCC-99-6-1999, (1999).

28 M.C. Rivara, "Selective refinement/derefinement algorithms for sequences of nested triangulations", *Int. J. Num. Meth. Engng.*, Vol. **28**, pp. 2889–2906, (1989).

29 M.C. Rivara, "Mesh refinement based on the generalized bisection of simplices", *SIAM J. Num. Anal.*, Vol. **2**, pp. 604–613, (1984).

30 M.C. Rivara y C. Levin, "A 3-D refinement algorithm suitable for adaptive and multi-mesh techniques", *Comm. in Appl. Num. Meth.*, Vol. **8**, 281–290, (1992).

31 M.C. Rivara, "New mathematical tools and techniques for refinement and/or improvement of unstructured triangulations", *5th International Meshing Roundtable'96*, Sandia Report SY96-2301, UC405, pp. 77–86, (1996).

32 K. Weiler, "Edge-based data structures and concepts for solid modeling in curved-surface environments", IEEE 0272-1716/85/0100-0021, (1985).

APÉNDICE

Configuraciones de tetraedros según el grafo orientado del 1-esqueleto

La Figura 16 muestra todas las posibles configuraciones para el grafo orientado basado en el 1-esqueleto de los tres tipos distintos de tetraedros. A la izquierda se muestra el tetraedro de referencia abierto y a la derecha su grafo correspondiente. La arista *e1* representa la *arista de referencia* y en cada cara del triángulo la arista mayor se indica por una línea de puntos.

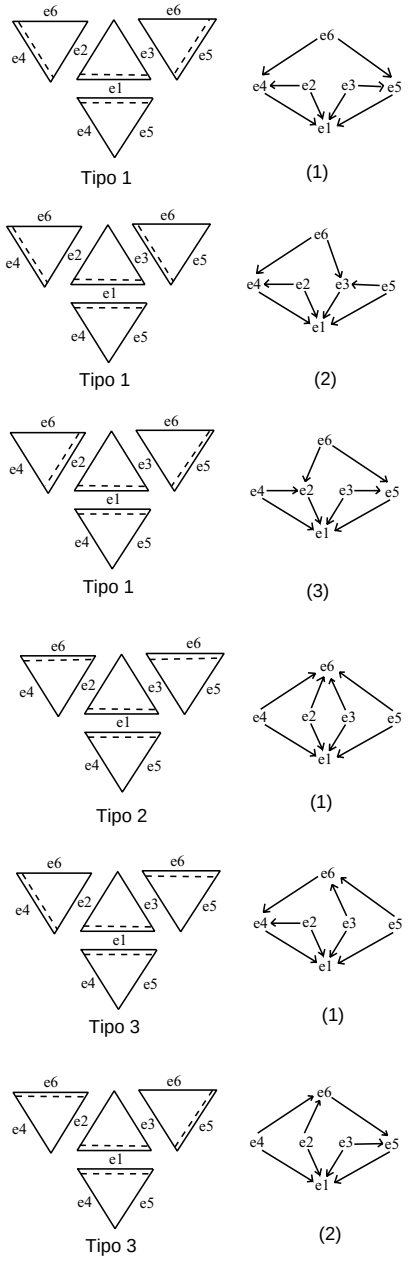


Figura 16. Configuraciones de G^1 para los tres tipos de tetraedros