UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS MATEMÁTICAS Y DE LA ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA

UN ALGORITMO DE PARTICIÓN DE REGIONES PLANAS EN REGIONES SIMPLEMENTE CONEXAS PARA GENERAR MALLAS ESTRUCTURADAS POR BLOQUES

TESINA
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS

PRESENTA:
GUSTAVO ADOLFO GARCÍA CANO

PABLO BARRERA SÁNCHEZ
FACULTAD DE CIENCIAS

CIUDAD DE MÉXICO A 19 DE SEPTIEMBRE DE 2016

Índice general

Introducción

En diversos problemas numéricos, la región de estudio está compuesta por un cuerpo principal, apéndice(s) y agujero(s). Según sea el caso, el grado de complejidad puede variar; por ejemplo, el río Xingú[1] o las arterias de una mano. La parte del río Xingú que se muestra en la figura 1a tiene diversas islas que pueden ser consideradas agujeros para intereses de la región, y que generan partes muy delgadas de río. Por otro lado, la figura 1b muestra parte de las arterias de una mano; en este caso, no se observan cuerpos que puedan parecer agujeros, sin embargo la arteria se ramifica en diversas direcciones que se pueden pensar como apéndices. Estos tipos de regiones, al utilizar mallas estructuradas, suelen presentar celdas elongadas, por lo que requieren ser modeladas adecuadamente.

(a) Río Xingú (b) Arterias de una mano

Figura 1: Regiones de estudio

Actualmente el modelo geométrico de las regiones de este tipo se hace utilizando mallas no estructuradas con métodos de elemento finito. En el presente trabajo se propone utilizar mallas estructuradas por bloques dividiendo la región de estudio mediante líneas de partición en subregiones simplemente conexas, en cada subregión crear una malla estructurada y conectarlas de manera adecuada.

Esto supone resolver tres problemas esencialmente:

1. Crear una representación de la frontera de la región y sus líneas de partición.

2. Identificar cada subregión (bloque) dada la información anterior.

[1]El río Xingú es un largo río amazónico brasileño, uno de los mayores afluentes de la vertiente meridional del río Amazonas, que discurre por los estados de Mato Grosso y Pará.

3. Generar de la malla en cada bloque y unirlos.

0.1. Antecedente

La línea de investigación que el grupo UNAMALLA ha seguido es la generación de mallas estructuradas ϵ-convexas sobre regiones simplemente conexas planas, utilizando la formulación variacional discreta [5]. Esta última plantea el problema de generar una malla estructurada sobre una región simple como la búsqueda de un homeomorfismo del cuadrado unitario a la región simple (Fig. 2); además, se busca este mapeo mediante un proceso de optimización de un funcional que garantice la obtención de una malla convexa en su óptimo, partiendo de una función inicial que se aproxime al mapeo. Se puede encontrar más información en los artículos [1–3].

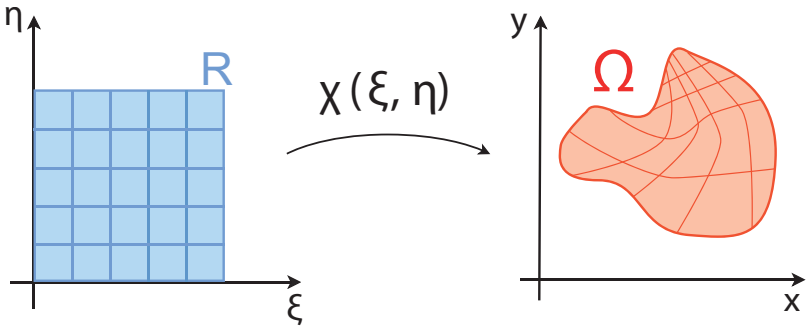


Figura 2: Homeomorfismo χ

Este trabajo ha desembocado en un software para la generación de mallas estructuradas: UNAMalla [2]. UNAMalla parte de una región poligonal simplemente conexa que representa la frontera de la región de estudio y ofrece herramientas para suavizar la frontera, determinar el mapeo en la frontera, generar la malla estructurada y ϵ-convexa y fijar puntos interiores de la malla.

En mi tesis de licenciatura [4], utilizando esta teoría, creé un programa para generar mallas estructuradas sobre regiones con agujeros. Si bien se logró este objetivo, en diversos ejemplos se apreciaron mallas con celdas elongadas que disminuyen su calidad, fenómeno que no es exclusivo de las mallas con agujeros. Era necesario buscar formas de evitarlo, en este sentido creé un programa que genera mallas estructuradas por bloque. El presente trabajo tiene como objetivo su descripción.

[2]Sitio Web de UNAMalla: `http://www.matematicas.unam.mx/unamalla`

Capítulo 1

Mallas estructuradas por bloques en regiones planas

1.1. Representación de la región

Siguiendo la misma línea del trabajo del grupo UNAMALLA, una región admisible puede ser una región poligonal simplemente conexa, orientada positivamente. Sin embargo esta definición no admite regiones con agujeros. Por lo que en este trabajo se utilizará una generación de la anterior dada en en el trabajo [4]:

Definición 1.1 (Regiones admisibles)**.** *Dadas:*

- *Una región poligonal simple $\Omega_{ext} \subset \mathbb{R}^2$, la cual se llamará "Región exterior".*
- *Un conjunto con cardinalidad $\rho \in \mathbb{N} \cup \{0\}$, compuesto por regiones poligonales simplemente conexas Ω_i con $i \in \{1, \ldots, \rho\}$. A cada una de éstas se le nombrará "agujero".*

Dónde los agujeros cumplen:

- *Ser ajenos:*
  $\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j$
- *Estar contenidos en el contorno exterior:*
  $\Omega_i \subset \Omega_{ext} \quad i \in \{1, \ldots, \rho\}$

Se llamarán "Región plana con ρ-agujeros admisible" o simplemente "Región con agujeros", a una región $\Omega \subset \mathbb{R}^2$ definida de la siguiente forma:

$$\Omega = \Omega_{ext} - (\bigcup_{i=1}^{\rho} \Omega_i^{\circ})$$

Esta definición permite la primera cuando ρ es cero y por lo tanto no hay agujeros.

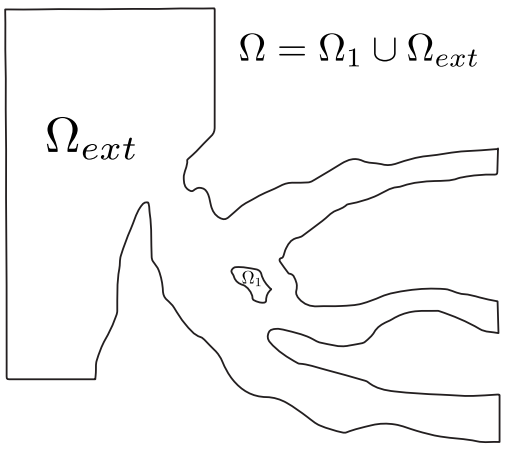


Figura 1.1: Ejemplo de región admisible

En el software las regiones poligonales son representadas utilizando la frontera, es decir, una región poligonal está dada por un conjunto ordenado y finito de vértices en el plano que representa su frontera. También necesitamos representar las líneas de partición, esto lo haremos también con regiones poligonales abiertas como sigue.

Definición 1.2 (Líneas de partición)**.** *Dada una región admisible* Ω*, se llamará "Líneas de partición" de* Ω *a una colección de trayectorias poligonales* T_i *con las siguientes características:*

- *El vértice final y el vértice inicial son parte de la frontera de* Ω *o de otra trayectoria.*
- *Todos los vértices de una trayectoria pertenecen al interior de* Ω *excepto tal vez el vértice inicial y/o el vértice final.*
- *La intersección entre dos trayectorias es a lo más un vértice, y ese vértice es inicial o final de una de las dos o de las dos.*

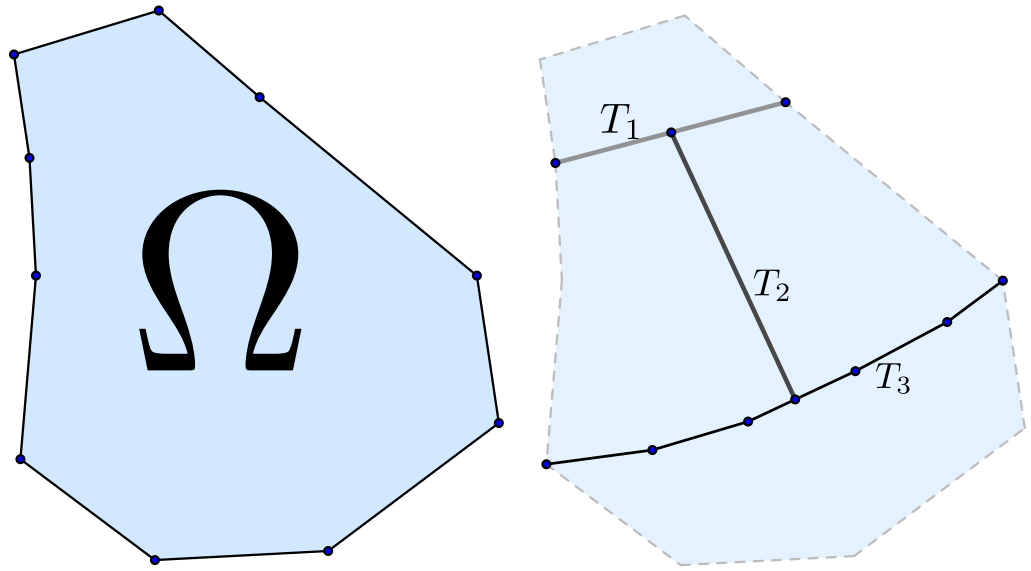


Figura 1.2: Ejemplo de líneas de partición sobre una región Ω

Estas líneas de partición son creadas dentro del programa de manera manual, al igual que los contornos de la región y sus agujeros. Un problema que está abierto es proponer métodos para generar una propuesta de líneas de partición. Aunque es claro que la elección de las líneas de partición está asociada a la geometría de la región y el problema que se quiera resolver.

1.2. Identificación cada bloque

Una vez que se tiene la región Ω y sus líneas de partición, hay que calcular las subregiones que inducen las líneas de partición. Para resolver este problema se diseñó un algoritmo, para el que se requiere presentar algunas definiciones y observaciones. Para facilitar la introducción de los conceptos, se empezará con regiones sin agujeros, y posteriormente con regiones con agujeros.

1.2.1. Definiciones

Definición 1.3 (Partición de una región simplemente conexa)**.** *Dada Ω una región poligonal simplemente conexa, se llamará "partición de Ω" a $\mathcal{P}$, una colección finita de regiones poligonales simplemente conexa ω_i, tales que:*

- $\omega_i^\circ \neq \emptyset \qquad \forall i$
- $\omega_i \cap \omega_j \subset \partial\omega_i \qquad \forall i \neq j$
- $\Omega = \bigcup_{i=0} \omega_i$

A cada región ω_i se les llamará "subregiones de Ω".

Nótese que la definición pide $\omega_i \cap \omega_j \subset \partial\omega_i \; \forall i \neq j$, por lo que podría suceder que $\omega_i \cap \omega j = \emptyset$ para algún o algunos índices i, j.

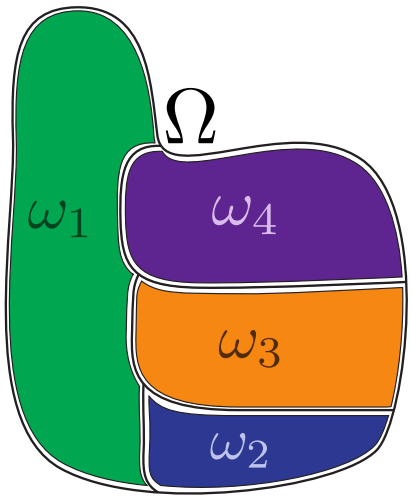


Figura 1.3: Partición de una región Ω en cuatro regiones

A continuación se presentan tres observaciones que serán de gran utilidad.

Sea Ω una región simplemente conexa, y $\mathcal{P}$ una partición de Ω.

Observación 1.1. *La orientación de $\partial\Omega$ induce una orientación sobre las fronteras de todas las subregiones, de tal forma que si la frontera de la región Ω es positiva, las fronteras de las subregiones tendrán la orientación positiva.*

Observación 1.2. *Sea ω_i una subregión de Ω en $\mathcal{P}$. Sean $\{\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_k}\}$ todas las subregiones de $\mathcal{P}$ aledañas a ω_i, es decir, $\omega_i \cap \omega_j \neq \emptyset \; \forall j \in \{i_1, i_2, \ldots, i_k\}$, entonces se pueden dividir a $\partial\omega_i$ en al menos k partes, de la siguiente forma:*

$$\mathcal{C}_\ell := \partial\omega_i \cap \omega_{i_\ell}$$

además:

$$\bigcup_{\ell=1}^{k} \mathcal{C}_\ell \subseteq \partial\omega_i$$

Cuando la igualdad no se da, existe una parte extra que pertenece a la frontera de Ω.

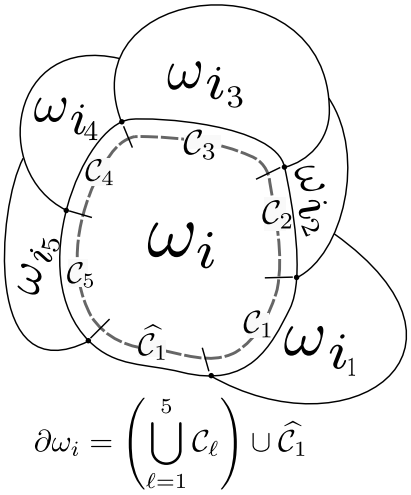


Figura 1.4: Ejemplo de curvas $\mathcal{C}_\ell$.

Observación 1.3. *Tomando los conjuntos de la observación 1.2 y usando la observación 1.1, cada subregión ω_i tiene una orientación, y ésta a su vez induce una orientación a la curva $\mathcal{C}_l$, y también ω_{i_ℓ} induce una orientación, y ésta es contraria a la que induce ω_i, es decir, si se recorren todas las fronteras de las subregiones ω_i con su orientación, entonces se recorren todas las curvas $\mathcal{C}_\ell$ dos veces, una vez en cada orientación.*

La definición de "partición de una región" (1.3) se deja intacta para regiones con agujeros, y por lo tanto las observaciones son iguales para las regiones con agujeros, tomando la orientación de los agujeros contraria a la orientación del contorno externo.

Dado que los conceptos que se han definidos no dependen del tipo de región que se use, a partir de ahora se denominará, de manera indiscriminada, "región" a una región poligonal simplemente conexa o a una región con agujeros.

Ahora se verá que las definiciones de "partición de una región" es equivalente a la definición de "líneas de partición de una región", es decir, que uno induce al otro y viceversa. El algoritmo que calcula cada subregión (o bloque) es el que muestra que las líneas de partición de una región inducen una partición de la región.

Lema 1.1. *Representación de una partición*
Dada una región Ω.
Una partición $\mathcal{P}$ induce un conjunto de líneas de partición y viceversa.
Por lo que estas dos definiciones son equivalentes.

Demostración. Sólo se hará la ida, ya que el regreso será demostrado con el algoritmo. En esta demostración es necesario que las operaciones sean de curvas poligonales en curvas poligonales, por lo que la definición de las operaciones usada en conjuntos no será adecuada en los casos en que el conjunto resultante no sea un conjunto cerrado, basta con tomar la cerradura para garantizar que sea un polígono.
Sea $A := \{\omega \in \mathcal{P} | \omega \cap \partial\Omega \neq \emptyset\}$, se pueden enumerar los elementos de A (pues $\mathcal{P}$ es finito). Una vez enumerados, se toman en ese orden, llámese ω_i, y se crean las líneas de partición de la siguiente forma:

- Si $\partial\omega_i$ no contiene parte de alguna trayectoria de las líneas de partición, entonces se define una trayectoria como:

  $$T := \overline{(\partial\omega_i - \partial\Omega)}$$

  Esta trayectoria cumple con las condiciones, pues termina y empieza con vértices en la frontera de la región, y todos los demás vértices están totalmente contenidos en el interior de la región.

- En caso contrario, sean $T_1, T_2, \ldots, T_k$ las trayectorias que tienen intersección con $\partial\omega_i$, se define el conjunto:

  $$PT := \overline{\left(\partial\omega_i - \bigcup_{j=0}^{k} T_j\right)}$$

  Si $PT = \emptyset$ entonces la región ω_i ya fue cubierta. Si no, entonces se toman las trayectorias formadas por cada componente conexa de PT.

Una vez hecho esto, en cada región de A, se repite de manera análoga, ahora usando las regiones de $\mathcal{P}$ que son vecinas a las regiones de A, y así hasta que se hayan cubierto todas las fronteras de las regiones de $\mathcal{P}$.

□

1.2.2. Algoritmo de partición de regiones

El algoritmo resuelve el problema de pasar de un conjunto de líneas de partición a una partición, es decir: dada una región Ω y una colección de trayectorias como se establece en la definición 1.2, devolver una colección de regiones como se describen en el algoritmo 1.3 (Véase ejemplo en el Anexo).

Dada una región simplemente conexa Ω y un conjunto de líneas de partición T, el esquema general del algoritmo es el siguiente:

1. Generar una gráfica asociada a la región de Jordán Ω.

2. Recorrer la gráfica e identificar cada región ω_i de la partición (Creación de la partición de la región Ω).

3. Recorrer la región y asociar una gráfica de conexiones (Obtención de la gráfica de conexiones).

Generación de la gráfica asociada a la región Ω

El algoritmo se auxilia de una gráfica que representa las conexiones de la región Ω junto con las líneas de partición T, y permite tomar decisiones de manera económica. Esta gráfica se genera de la siguiente manera:

Sea G una digráfica etiquetada con conjunto de vértices V y conjunto de aristas E, definidos como sigue: $V := \{v_i\}$, donde v_i está biunívocamente asociado a un vértice de T que pertenece a la frontera de Ω o a más de una línea de partición de T.
Las aristas de E serán aristas dirigidas y etiquetadas de la siguiente forma:

- Si v_i y $v_j \in \partial\Omega_i \cap V$ y existe una ruta en la poligonal $\partial\Omega_i$ en la orientación de la misma que va de v_i a v_j sin pasar por otro vértice de T, entonces $\overrightarrow{(v_i, v_j)} \in E$ con la etiqueta $\{C, i\}$. Con el cuidado de $\Omega_0 := \Omega_{ext}$.

- Si v_i y $v_j \in T_i \cap V$ y existe una ruta en T_i que vaya de v_i a v_j (o viceversa, ya que T_i no está orientada), entonces $\overrightarrow{(v_i, v_j)} \in E$ y $\overrightarrow{(v_j, v_i)} \in E$, con la etiqueta $\{0, i\}$ cada una. Hay que observar que en este caso v_i y v_j, pueden o no pertenecer a la frontera de Ω, y además esta arista es diferente a la asignada con al regla anterior por el etiquetado — Esto significa que existen casos en los que hay ambas aristas y de aquí la necesidad de etiquetar las aristas y pode diferenciarlas.

Las etiquetas en las aristas se debe a que queremos aprovechar el conocer la orientación de las aristas en la frontera de Ω y además queremos diferenciar entre una arista en la frontera que una v_i con v_j y una arista en el interior los mismos nodos.

El algoritmo que crea la arista sigue el pseudocódigo del algoritmo 1.

Creación de la partición de la región Ω

Dada una región de Ω junto con un conjunto de líneas de partición T, y generando la gráfica $G := (V, E)$ como en la sección anterior, ahora se puede identificar cada región ω_l del conjunto de partición como sigue:

Se toma $v_i \in V$ que está en $\partial\Omega$, entonces v_i tiene sólo una arista de salida con la etiqueta ("C",j), que conecta a v_i con v_k, se recorre Ω_j desde v_i hasta v_k agregando cada vértice de este recorrido en la región ω_l. Cuando se llega a v_k, se quita la arista que se acaba de usar, y se toman todas las aristas en G que salen de v_k, nombrándolas "aristas e_m", y a los nodos a los que llegan v_{k_m}. Cada arista e_m representa una trayectoria sobre la frontera de Ω o una línea de partición de T. Se mide entonces el ángulo a_{k_m} que hay en v_k al recorrer desde v_i hasta v_{k_m} sobre las trayectorias en la frontera Ω y las líneas de partición T, según sea el caso. Se toma el ángulo $a_{\widehat{k_m}}$ mínimo. El algoritmo sigue de manera recursiva cambiando i por k, k por $\widehat{k_m}$, agregando a la región ω_l hasta que se llegue al primer vértice. Finalmente se cambia de nodo para empezar esta regla desde el principio.

Una vez recorridos todos los nodos $v_i \in V$ que están también $\partial\Omega$, se recorren ahora sólo los nodos que no están en la frontera de Ω. Se repite el algoritmo anterior en ellos, salvo que se debe elegir la primer arista que se toma en cada nodo. Para elegir una primer arista basta con buscar entre las aristas que salen del nodo v_i cual de ellas no tiene arista de regreso; si alguna no tiene arista de regreso, se debe tomar como primer arista a recorrer y seguir el algoritmo anterior. Siempre existe al menos un nodo que

cumpla con esta condición, pues al haber ya recorrido al menos una región, entonces se quitaron las aristas recorridas. También se deben de quitar los nodos de la gráfica que ya no tienen aristas de salida. Al final se obtiene una gráfica vacía.

A continuación se describirán unas funciones sencillas, de las que sólo se explicará lo que hacen. Las funciones más complicadas se detallan en pseudocódigo al final de la sección.

aristaenfrontera Dado un *nodo* $\in G$, devuelve la primera arista de G que sale de *nodo* y pertenece a $\partial\Omega$.

aristasSalen Dado un *nodo* $\in G$, regresa la lista de aristas de G que salen de *nodo*.

buscarCamino Dado un *nodo* $\in G$, devuelve la primera arista de G que sale de *nodo* y no tiene arista de regreso.

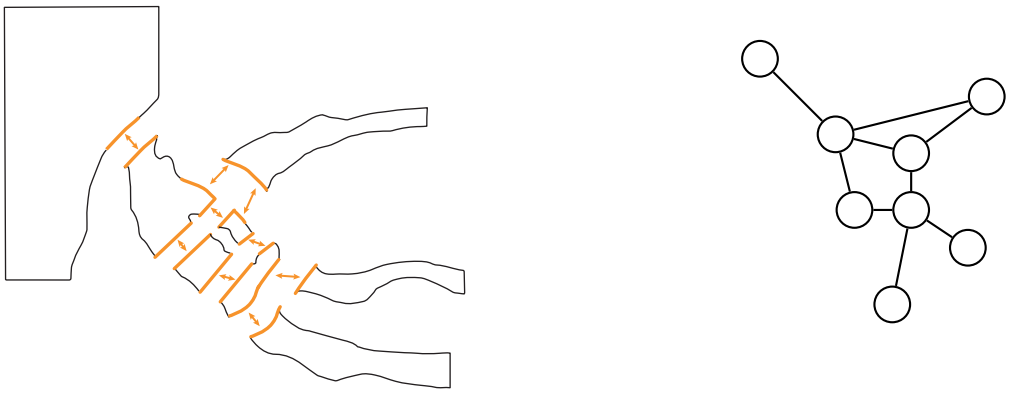
generarListadeNodosParticion Dada una gráfica G, da la lista de nodos de G que no pertenecen a $\partial\Omega$.

penultimoenlaArista Dada una *arista* $\in G$ dirigida, regresa el penúltimo vértice en la poligonal que representa a la arista (ya sea en $\partial\Omega$ o en una línea de partición).

penultimoenlaArista Dada una *arista* $\in G$ dirigida, regresa el segundo vértice en la poligonal que representa a la arista.

Con esto, se genera una partición de Ω dado un conjunto de líneas de partición. Y queda demostrada la equivalencia de definiciones.

Obtención de la gráfica de conexiones



(a) Región con agujero partida (método de malla estructurada con agujeros)

(b) Gráfica asociada a la región con agujero partida

Figura 1.5

Finalmente es necesario tener una representación adecuada de la región Ω como unión de subregiones, es decir, conocer las conexiones entre cada ω_i, para esto se hizo un pequeño algoritmo extra. Si se observa en la línea 4 del algoritmo 3 y las líneas 9, 13, 18, 26 y 31 del algoritmo 4, a cada vértice en las poligonales de Ω y T, cuando se crea la región ω_i, se les añade la información del índice al que pertenecen, por lo

que basta con recorrer todos los vértices de las poligonales tanto de Ω como de T para saber a qué regiones pertenecen los vértices, y cuando haya más de una región ω_i al que pertenece un vértice, entonces hay una conexión entre estas regiones, con lo que se puede formar una gráfica $H := (V, E)$ con conjunto de vértices $V := \{i|$ existe $\omega_i \in P\}$ y $E := \{\{i, j\}|\exists v \in \partial\Omega$ y $v \in \omega_i \cap \omega_j\}$. Esta gráfica es una representación de

la región Ω visto como uniones de regiones ω_i.

Algoritmo 1: Algoritmo para la generación de la gráfica asociada a la región.

```
Data: Ω una poligonal simplemente conexa y T un conjunto de líneas de
      partición
Result: G := (V, E) una digráfica etiquetada
1  foreach Ω_i ∈ Ω do
2      foreach n_j ∈ Ω_i do
3          inicio = NULL;
4          fin = NULL;
5          ultimoInicio = NULL;
6          ultimoFin = NULL;
7          if n_j tiene más de dos vecinos then
8              agregar n_j a V;
9              if ultimoInicio == NULL then
10                 ultimoInicio = n_j;
11             else
12                 ultimoFin = n_j;
13             if inicio == NULL then
14                 inicio = n_j;
15             else
16                 fin = n_j;
17                 agregar {(inicio, fin), ("C",i)} a E;
18     if ultimoInicio != NULL y ultimoFin != NULL then
19         agregar {(ultimoInicio, ultimoFin), ("C",i)} a E;
20 foreach T_i ∈ T do
21     foreach n_j ∈ T_i do
22         inicio = NULL;
23         fin = NULL;
24         if n_j tiene más de dos vecinos then
25             agregar n_j a V;
26             if ultimoInicio == NULL then
27                 ultimoInicio = n_j;
28             else
29                 ultimoFin = n_j;
30             if inicio == NULL then
31                 inicio = n_j;
32             else
33                 fin = n_j;
34                 agregar {(inicio, fin), (0,i)} a E;
35                 agregar {(inicio, fin), (0,i)} a E;
```

Algoritmo 2: Algoritmo para la generación de la gráfica asociada a la región

Data: Ω una poligonal simplemente conexa y T un conjunto de líneas de partición, $G := (V, E)$ una digráfica etiquetada
Result: Conjunto P de regiones ω_i

```
1  bool hayacamino=true;
2  int i;
3  while hayacamino do
4      foreach nodo ∈ G do
5          arista = aristaenfrontera(nodo);
6          if arista ≠ null then
7              ω = recorrerRegión(nodo,arista);
8              P.agregar(ω);
9          else
10             arista = buscarCamino(nodo);
11             if arista ≠ null then
12                 ω = recorrerRegión(nodo,arista);
13                 P.agregar(ω);
14             else
15                 hayacaminos = false;
16 i=0;
17 nodoParticion = generarListadeNodosParticion(G);
18 while nodoParticion.size()> 0 do
19     nodo = nodosParticion[i];
20     aristas = aristasSalen(nodo);
21     if aristas.size() == 1 then
22         arista = aristas.first();
23         ω = recorrerRegión(nodo,arista);
24         P.agregar(ω);
25     else if aristas.size() ¿ 1 then
26         arista = buscarCamino(nodo);
27         if arista ≠ null then
28             ω = recorrerRegión(nodo,arista);
29             P.agregar(ω);
30         i++;
31     else
32         nodoParticion.quitar(nodo);
```

Algoritmo 3: Función recorrerRegión

```
Data: nodo: vértice o nodo de una gráfica G, arista: arista inicial de G que
      sale de nodo.
Result: ω región polinonal
1  aristaAnterior = arista;
2  p1 = nodo;
3  ω.agregar(p1);
4  p1→addRegion(regions.size());
5  recorrerArista(arista,·ω,P.size()); G.quitarArista(arista); p2 = arista.final();
6  repeat
7      pt2 = p1;
8      pt3 = penultimoenlaArista(aristaAnterior);
9      p1 = p2;
10     aristas = aristasSalen(p1); if aristas.size() == 1 then
11         arista = aristas.first();
12         p2 = arista.final();
13         recorrerArista(arista,ω,P.size());
14         graphC.removeArista(arista);
15         aristaAnterior = arista;
16     else if aristas.size()>0 then
17         j=0;
18         while j != aristas.size() do
19             if aristas[j].final() == pt2 and !aristas[j] ∈ ∂Ω and !aristaAnterior
                ∈ ∂Ω and aristas[j].getNumero() == aristaAnterior.getNumero()
               then
20                 aristas.quitar(j);
21             else
22                 j++;
23         if aristas.size()>0 then
24             pt = segundoenlaArista(aristas.first());
25             areamin = angleBetween2Segments(pt3,p1,pt);
26             idxMin = 0;
27             for j =1; j<aristas.size(); j++ do
28                 pt = segundoenlaArista(aristas[j]);
29                 area= angleBetween2Segments(pt3,p1,pt);
30                 if areamin>area then
31                     areamin = area;
32                     idxMin = j;
33             arista = aristas[idxMin];
34             p2 = aristas[idxMin].final();
35             recorrerArista(arista,ω ,P.size());
36             graphC.removeArista(arista);
37             aristaAnterior = arista;
38 until p1 == p2 or p2 == nodo;
```

Algoritmo 4: Función recorrerArista

Data: *arista*: arista de G. ω región a la que se le agregaran vértices en arista.
r: índice del ω_r al que pertenece.

```
1  j = arista.getNumero();
2  if arista.enContorno() then
3      ini = arista.inicio().number();
       ;        // number se refiere al índice de la región ω a la que
        pertenece
4      fin = arista.final().number();
5      if ini ¿ fin then
6          for i=ini+1; i<sizeof(Ω_j); i++ do
7              v = Ω_j.vertice[i];
8              ω.append(v));
9              v.addRegion(r);
10         for i=0; i ¡= fin; i++ do
11             v = Ω_j.vertice[i];
12             ω.append(v));
13             v.addRegion(r);
14     else
15         for i=ini+1; i<=fin; i++ do
16             v = Ω_j.vertice[i];
17             ω.append(v));
18             v.addRegion(r);
19 else
20     ini = arista.inicio().number(j);
       ;   // number se refiere al índice de la línea de partición T a
        la que pertenece.
21     fin = arista.final().number(j);
22     if ini ¡ fin then
23         for i = ini + 1; i ≤ fin; i + + do
24             v = T_j.vertice[i];
25             ω.append(v);
26             v.addRegion(r);
27     else
28         for i = ini − 1; i ≥ fin; i − − do
29             v = T_j.vertice[i];
30             ω.append(v);
31             v.addRegion(r);
```

Algoritmo 5: Función angleBetween2Segments

Data: a, b, c: vértice de una poligonal
Result: $angle$: ángulo $\measuredangle abc$

```
1  eps = 1.11e − 16;
2  vx = a.x() − b.x();
3  vy = a.y() − b.y();
4  wx = c.x() − b.x();
5  wy = c.y() − b.y();
6  if ( |vx| < eps and |vy| < eps ) or ( |wx| < eps and |wy| < eps ) then
7      angle = 0;
8  else
9      det = vx ∗ wy − vy ∗ wx;
       ;   // Hay que recordar que en la computadora los ejes tiene una
        reflexión en X
10     prod = vx ∗ wx + vy ∗ wy;
11     angle = atan2(det, prod);
12     if |angle| > eps then
13         if angle < 0 then
14             angle = 2 ∗ π + angle;
```

1.3. Continuidad de mallas estructuradas por bloques

Con el método anterior se ha generado un conjunto de subregiones a partir de una región Ω con líneas de partición (Fig. 1.6).

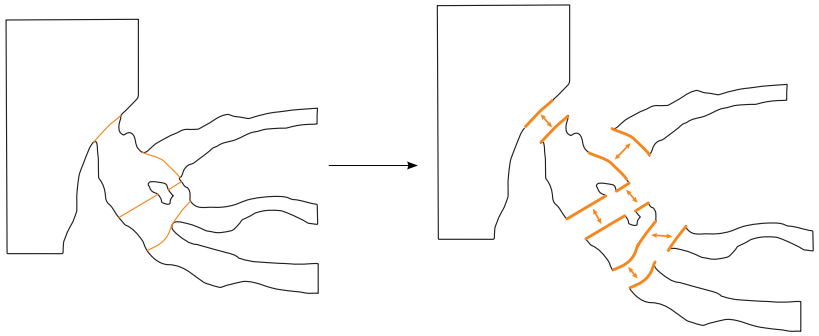


Figura 1.6: Región Ω separada en subregiones ω_i.

En cada región ω_i hay que generar una malla estructurada, sin embargo, si estas mallas se generan arbitrariamente es común que la unión sea discontinua (Fig. 1.7).

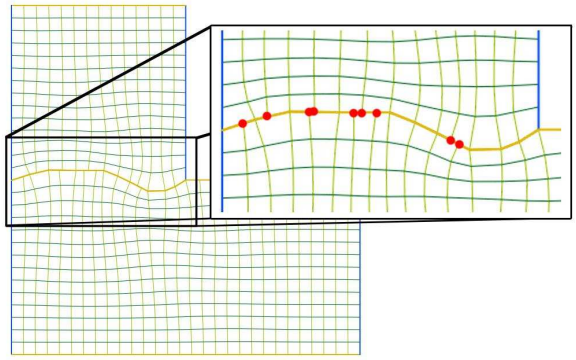


Figura 1.7: Malla estructurada por bloques discontinua.

Para evitar esto, se ha creado un método para forzar la continuidad (Véase un ejemplo en el Anexo A.4).

El método de generación de mallas desarrollado por el grupo UNAMALLA, busca un mapeo discreto del cuadrado unitario a la región de estudio(Fig. 1.8). Una condición para buscar el mapeo es definirlo sobre las fronteras, es decir, primero hay que definir los puntos de la malla sobre la frontera. Gracias a esto, se pueden definir distintas distribuciones de puntos sobre una misma elección de fronteras, como los ejemplos de la figura 1.9.

Al poder elegir los puntos sobre las cuatro fronteras de cada región, se selecciona una distribución *ad hoc* de los puntos para que al final la unión de los bloques sea continua. Esto también se vuelve una limitante, pues en este tipo de mallas ya no es sencillo definir distribuciones de puntos sobre los bloques (como en la figura 1.9). Aunque esto es un costo aceptable en la mayoría de los casos.

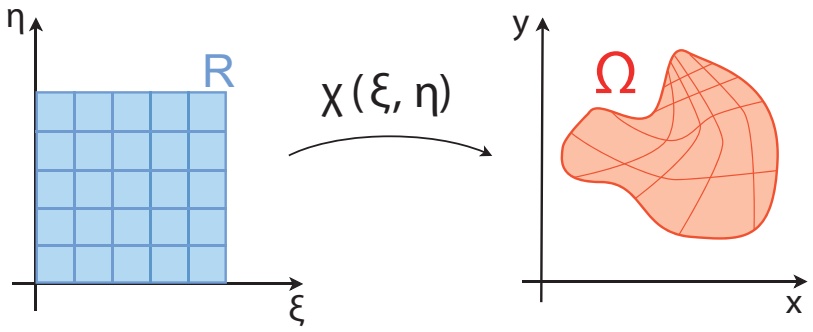


Figura 1.8: Homeomorfismo χ

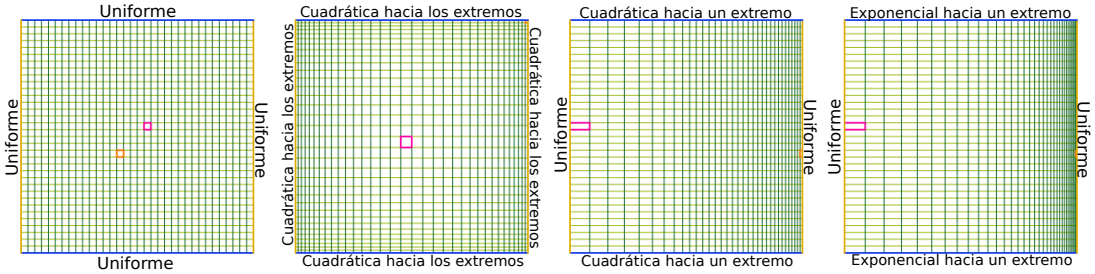


Figura 1.9: Cuatro distribuciones de puntos sobre la frontera

1.3.1. Método

El método consta en esencia de tres partes: el **preprocesamiento**,el **procesamiento**, y el **mallado**. El primero acomoda los datos para poder tomar decisiones; el segundo se encarga de tomar las decisiones para forzar la continuidad; y el tercero genera la malla en cada bloque. La idea principal de este método consiste en elegir una regla de orden (o prioridad) para las cuatro fronteras de cada bloque, pensando en cada frontera como una curva homeomorfa a un intervalo cerrado, y llenarlas en ese orden, y no bloque a bloque.

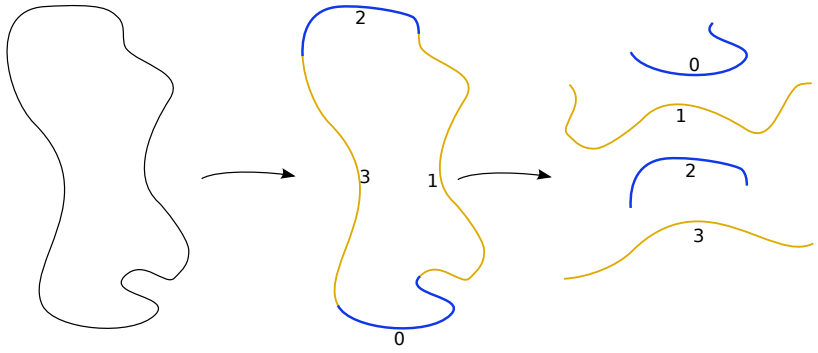


Figura 1.10: Frontera de una región vista como 4 curvas fronteras

Preprocesamiento

Se toma una subregión ω_i. Se ve a $\partial\omega_i$ como una colección de 4 fronteras (las fronteras asignadas a las cuatro fronteras del cuadrado unitario). Ahora tómese el

conjunto $\mathcal{F}$ como la unión de las colecciones de 4 fronteras de todas las subregiones, es decir, la unión de todas las fronteras de todos los bloques.
Todos los elementos de $\mathcal{F}$ son curvas homeomorfas a un intervalo cerrado de $\mathbb{R}$, por lo tanto es posible calcular su longitud. Ordénese el conjunto $\mathcal{F}$ por su longitud de curva de menor a mayor. Dentro del sistema, el conjunto $\mathcal{F}$ está formado por elementos que son conjuntos ordenados, cada elemento está formado de la siguiente manera:

$$(id, (ini, fin), long, i)$$

donde i es el índice del bloque ω_i al que pertenece, $long \in \mathbb{R}$ es la longitud de la curva, $id \in \{0, 1, 2, 3\}$ es el número que identifica a la curva frontera, (ini, fin) son los índices sobre $\partial\omega_i$ donde inicia y termina la curva frontera, respectivamente. Estos elementos tienen una función de orden asociada con la tercera coordenada (la de longitud).

Procesamiento

Con las curvas frontera ordenadas se debe definir la distribución de los puntos sobre las mismas. Para esto se tienen, *grosso modo*, las siguientes reglas:

- Se recorre $\mathcal{F}$ de menor a mayor.
- Si la curva no tiene intersección con otras curvas, se distribuyen de manera uniforme los puntos sobre ésta.
- Si la curva sí tiene intersecciones con otras curvas, se separan las zonas con intersección en dos grupos: los intervalos con puntos ya asignados por la intersección y los intervalos que no tienen puntos asignados por alguna intersección, o no tiene intersección. Estos grupos hacen una partición de la curva frontera, se puede ordenar usando la orientación de la propia curva. La idea es que en los intervalos sin puntos asignados, éstos se asignan con una distribución uniforme (sobre ese intervalo), y donde ya se tienen puntos asignados por otra curva frontera, se toman exactamente esos puntos para asignarlos. Es importante hacer notar que en el caso de que haya algún intervalo de intersección con otra curva frontera que no ha sido asignada, significa que ésta tiene una longitud mayor, y por lo tanto después se analizará y se le asignarán los puntos que se calculan en este paso. Esto significa que el método está bien definido.

Esta última parte hay que explicarla mejor. Cuando se tienen todos los puntos ya asignados, simplemente se toman todos las intersecciones y se unen en la orientación de la frontera. Es importante revisar que ésta tenga la cantidad de puntos que pide el usuario. Cuando se tienen intervalos con puntos asignados y otros intervalos donde no, entonces se sigue la siguiente regla:
Primero hay que analizar la estructura de la curva frontera:

- Se buscan las intersecciones de la curva con otros bloques auxiliándose de la gráfica generada en el método anterior 1.2.2.
- Se ordenan las intersecciones con respecto a la orientación de la frontera.
- Se juntan las zonas sin definir y se ordenan (en el sentido de la orientación de la curva).

Después, se suma la cantidad de puntos que haya en los segmentos ya asignados, se restan los puntos en común (cuando hay segmentos contiguos), se mide la longitud de los segmentos libres y se suman.
Luego, se calcula la cantidad de puntos restantes para completar la cantidad de puntos que pide el usuario en esta frontera. Se pide que al menos reste 1 punto para poder seguir.
Se calcula cuántos puntos se deben repartir en cada segmento libre, tomando su longitud de tal modo que se distribuyan los puntos lo más uniforme posible sobre los segmentos libres. Si al final faltan puntos para distribuir, éstos se asignan en el segmento libre más largo.
A continuación se toman lo segmentos ya asignados y libres, según corresponda el caso, y se asignan los puntos para la frontera, ya sea del bloque adyacente o se distribuyen uniformemente (según la cantidad calculada anteriormente, tomando en cuenta que los extremos ya están asignados por los segmentos adyacentes, cuando los hay). Los casos están en la figura 1.11.

Para encontrar las intersecciones con los bloques adyacentes, y cuando se toman los puntos sobre los segmentos ya asignados, hay que tomar en cuenta que estos puntos han sido calculados, por lo que puede tener errores numéricos al momento de querer comparar dos puntos.

Se creó una estructura de datos que permite guardar cuatro colecciones de puntos, cada una para una de las curvas fronteras de un bloque (*boundaryContour*). Con una lista de esta estructura, tenemos todas las fronteras en su orden original. Cuando se termina el proceso anterior, podemos revisar que sean coherentes, es decir, que las fronteras 0 y 2 tengan la misma dimensión, al igual que las fronteras 1 y 3. Esta estructura ayuda a generar los contornos poligonales de cada bloque ω_i con los puntos del mapeo ya asignados.

Mallado

Finalmente, cada malla se genera con un método de interpolación transfinita utilizando los contornos poligonales calculados en la parte anterior, posteriormente se utiliza un funcional para optimizar bloque a bloque (este proceso es perfectamente paralelizable), y se termina con una malla convexa y continua.

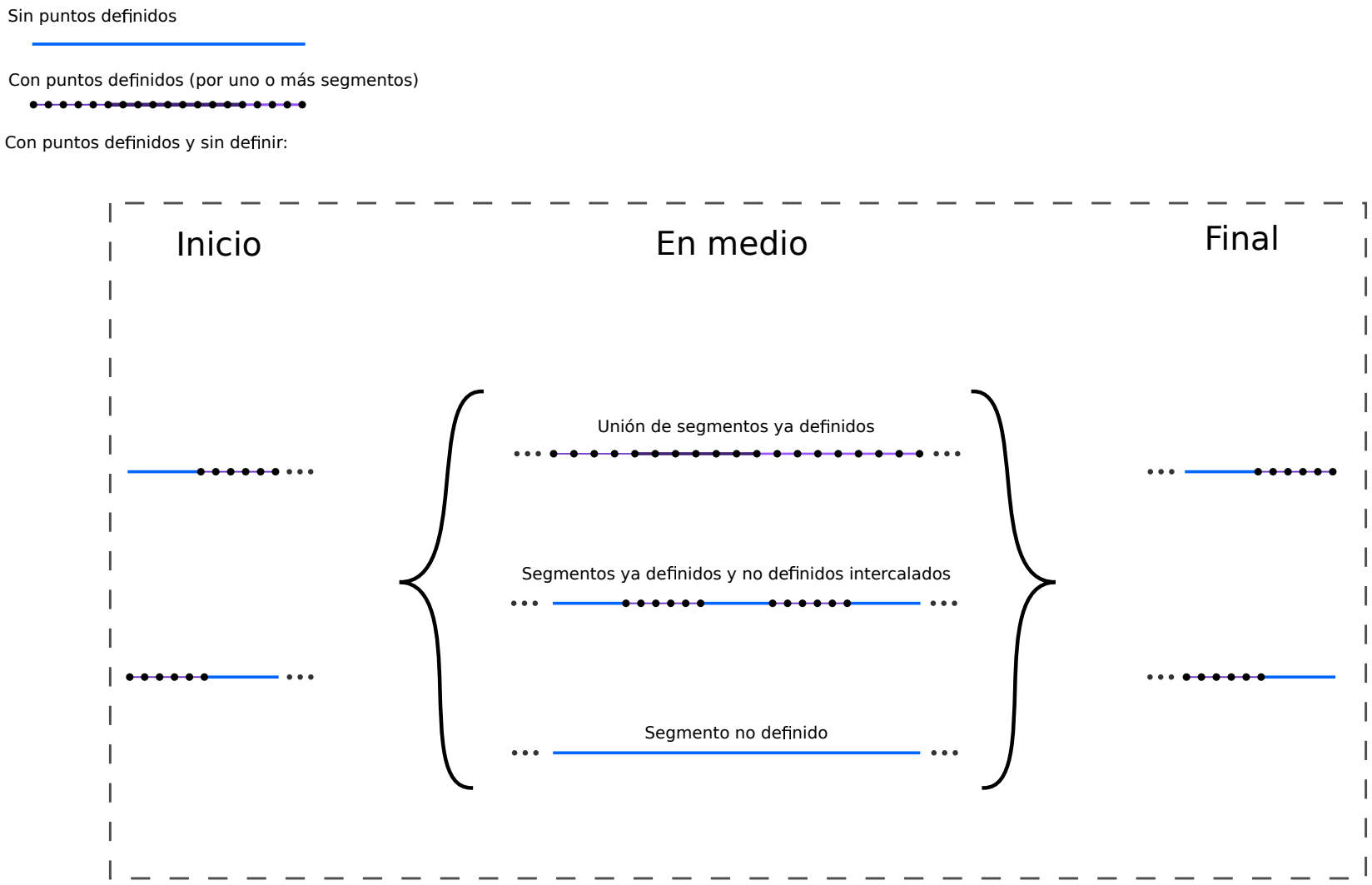


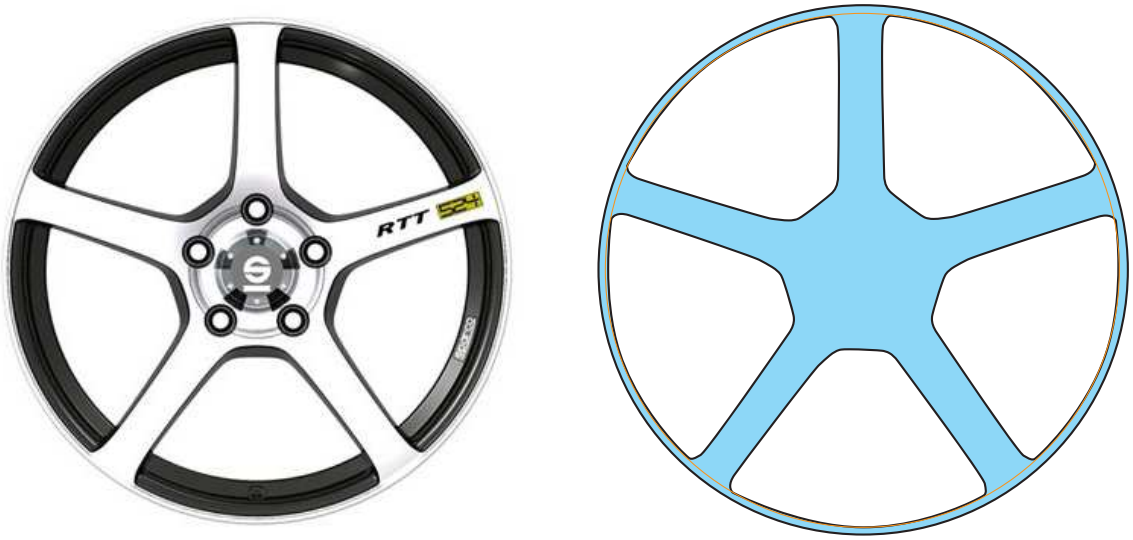
Figura 1.11: Casos de distribuciones de segmentos en una curva frontera

Apéndice A

Ejemplo: Rin de automóvil

A.1. Definición del dominio

Es posible definir el espacio de trabajo basándose en una imagen. En este caso, se partió de la imagen A.1a de un rin de automóvil. A partir de ésta se generó un modelo plano que representa la región en la que se va a generar una malla. Este modelo (Figura A.1b) representa la región de estudio.

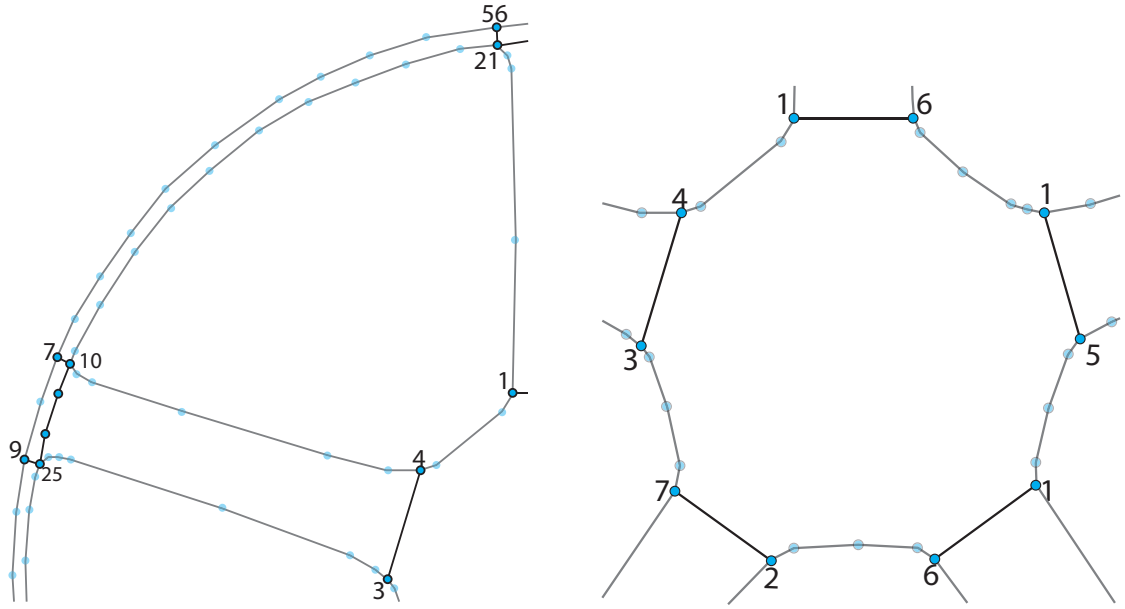


(a) Rin de automóvil (b) Modelo plano del rin

Figura A.1

Se puede ver que está compuesta por un círculo y cinco agujeros, que se puede representar con las siguientes regiones poligonales simples Ω_{ext}, Ω_1, Ω_2, Ω_3, Ω_4, Ω_5, como se ve en la figura A.3.

Una vez definidas estas regiones poligonales simples, se define la región de estudio como $\Omega := \Omega_{ext} - \left(\bigcup_{i=1}^{5} \Omega_i\right)$. Es necesario separarla en regiones poligonales simples que, si tienen intersección, la tienen sólo dentro de la frontera. Esto se hace mediante líneas de partición. Para este caso se asignaron 20 líneas de partición, las cuales están distribuidas de la siguiente forma: tres líneas para separar cada barra del rin de la orilla (Fig. A.2a), que en total suman quince, y cinco para separar cada barra del centro (Fig. A.2b).



(a) Región Ω (b) Región Ω

Figura A.2: Líneas de partición

Se puede ver en la figura A.4 por separado cada línea de partición T_i con el orden de los puntos que las componen.

Esto concluye la definición del dominio, en este ejemplo el dominio es la región Ω junto con sus líneas de partición.

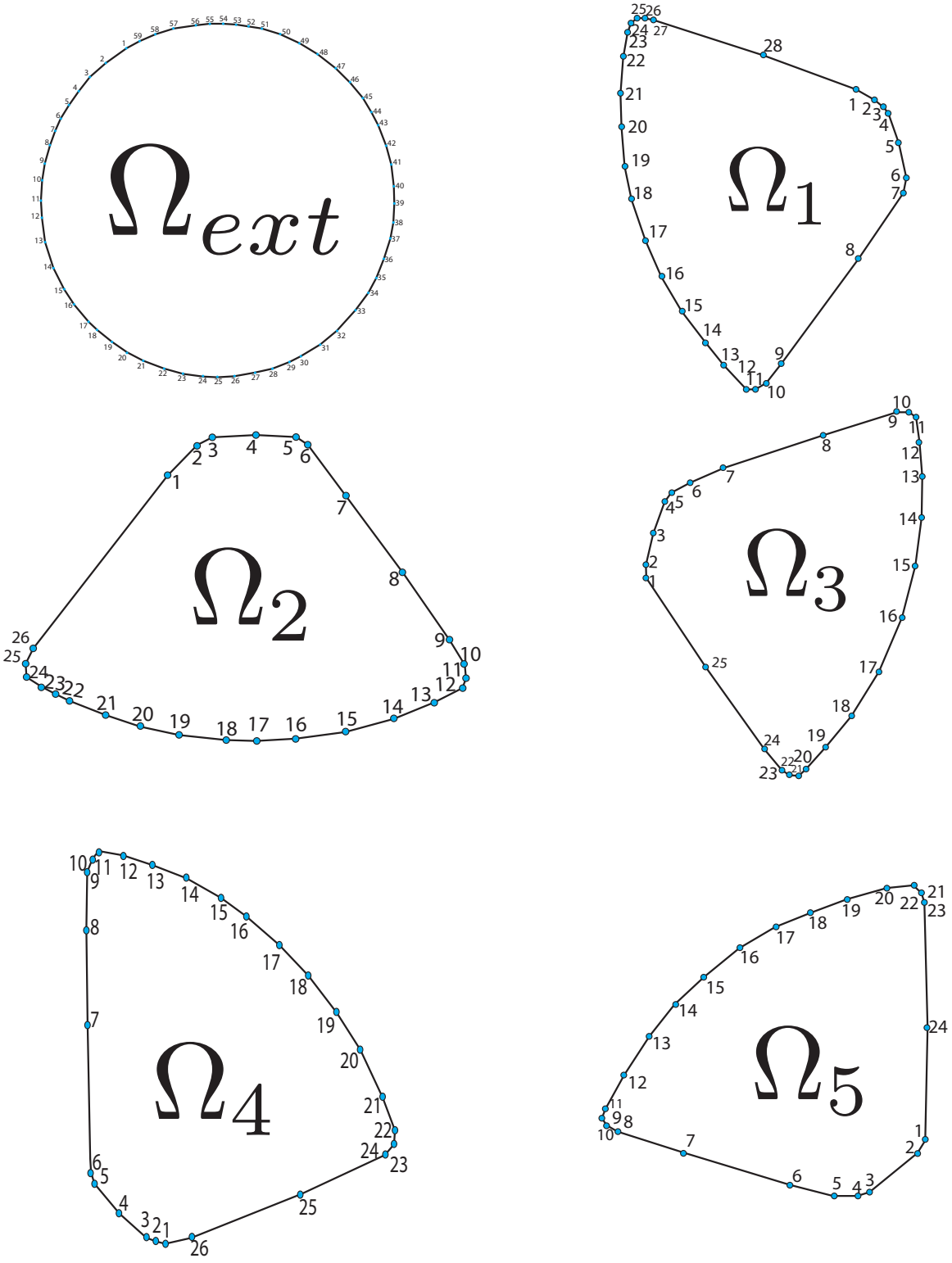


Figura A.3: Región Ω

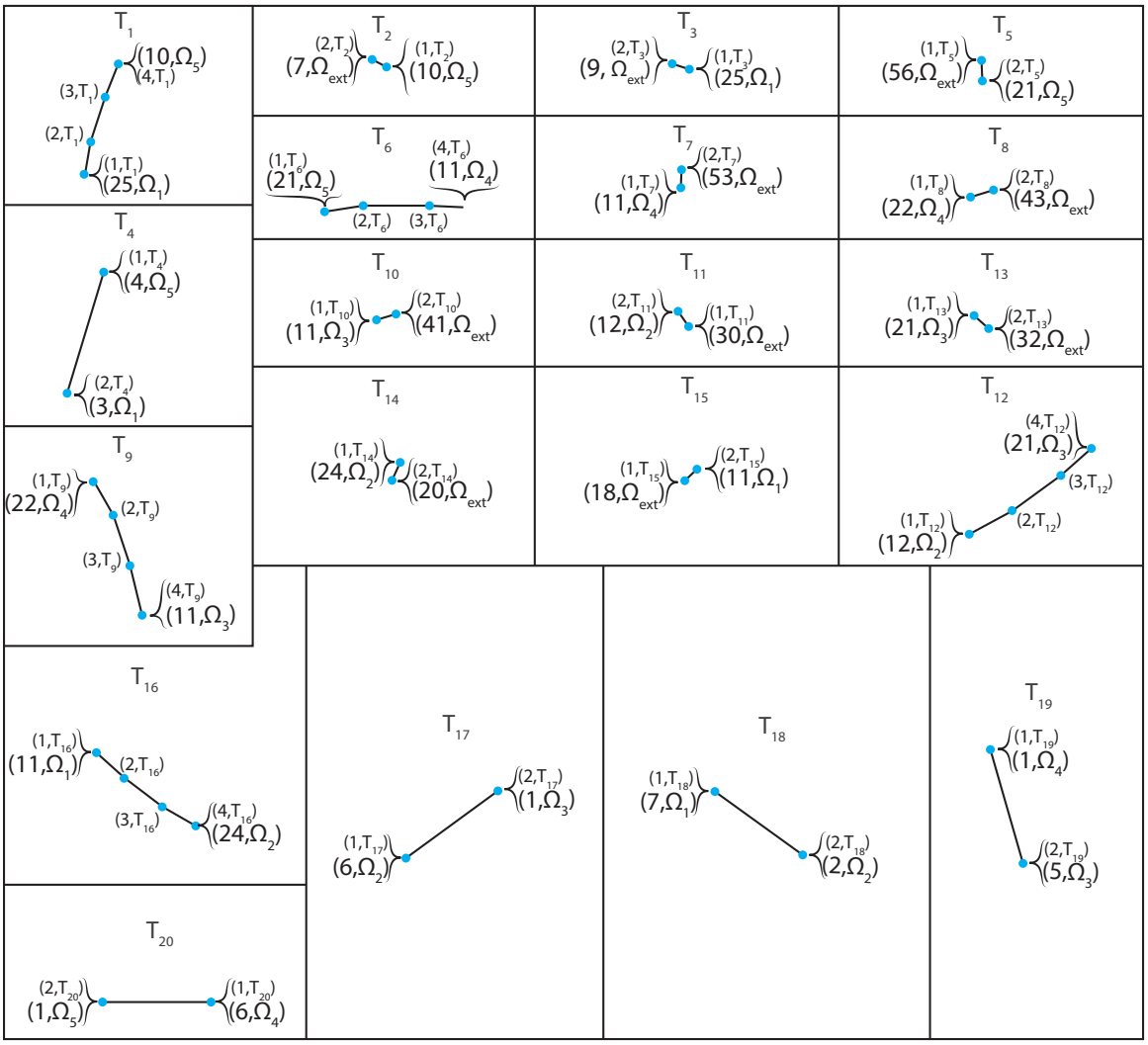
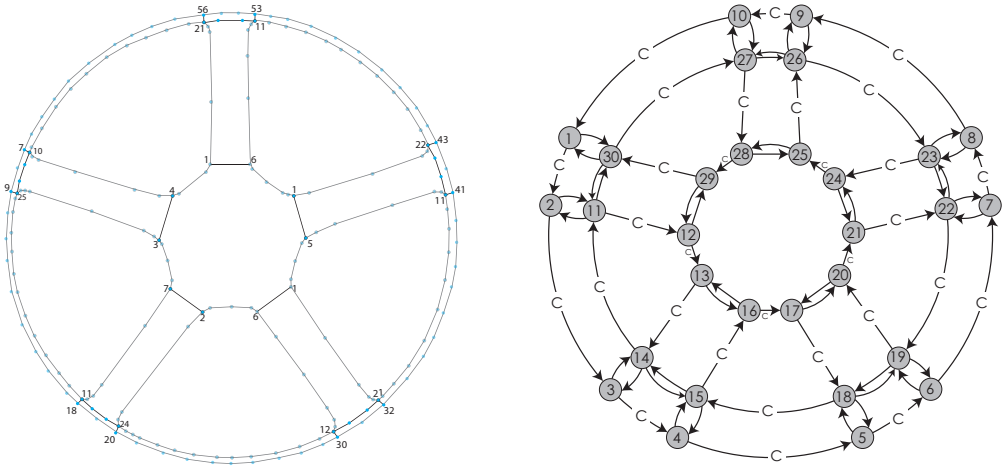


Figura A.4: Lista de lineas de partición T_i

A.2. Algoritmo de partición

Primero se debe construir la gráfica asociada a la región. Se construye con los puntos de la región que tienen más de dos vecinos (es importante tener presenta que los vecinos se cuentan incluyendo las líneas de partición), y asignando las aristas como se ve en la figura A.5b.



(a) Región Ω con su conjunto de líneas de partición

(b) Gráfica etiquetada asociada a Ω y su conjunto de líneas de partición.

Figura A.5

Es importante observar que el orden de los nodos de la gráfica se hereda del orden de sus vértices asociados en la región poligonal y el orden de las regiones (Ω_1, Ω_2, etc.). Además que la región Ω_{ext} se toma como Ω_0 (i.e. $\Omega_0 := \Omega_{ext}$).

A continuación se describe paso a paso el algoritmo que identifica la partición de la región de estudio, basándose en el que ya fue descrito en el capítulo 1.2.2. Es iterativo, y el proceso principal consta de los siguientes pasos:

1. Se toma un nodo inicial n_i, según el orden de los nodos en la gráfica.
2. Llamemos E_i al conjunto de aristas que salen del nodo n_i.
   - *a*) Si E_i contiene una arista etiquetada como contorno, se toma esta arista.
   - *b*) Si $\exists\{u, v\} \in E_i talque \{v, u\} \notin E$, con E el conjunto de aristas de la gráfica, entonces se toma esta arista {u,v}.
   - *c*) Si no, entonces se ignora este nodo n_i, se pasa al siguiente y se regresa al paso 1
3. La arista que se toma en el paso anterior es de la forma $\{n_i, n_j\}$. Hay que tomar ahora la tripleta $(n_j, \{n_i, n_j\}, \overline{E_j})$, donde $\overline{E_j}$ como el conjunto de aristas que salen de n_j restando la arista $\{n_j, n_i\}$. Con esta tripleta se calcula el conjunto de ángulos Θ correspondientes a los ángulos entre la poligonal en la región de estudio asociada a la arista $\{n_i, n_j\}$, y las poligonales en la región de estudio asociadas a las aristas del conjunto $\overline{E_j}$, calculando un ángulo por cada arista de $\overline{E_j}$, para luego elegir el ángulo menor de Θ y tomar la arista asociada a éste.

Es importante destacar que el ángulo se calcula con los segmentos dentro de las poligonales que inciden en el vértice asociado a n_j. Se continúa sustituyendo n_i con n_j y se avanza desde el paso 2, hasta que n_j sea el nodo del paso 1 (Ver figura A.6).

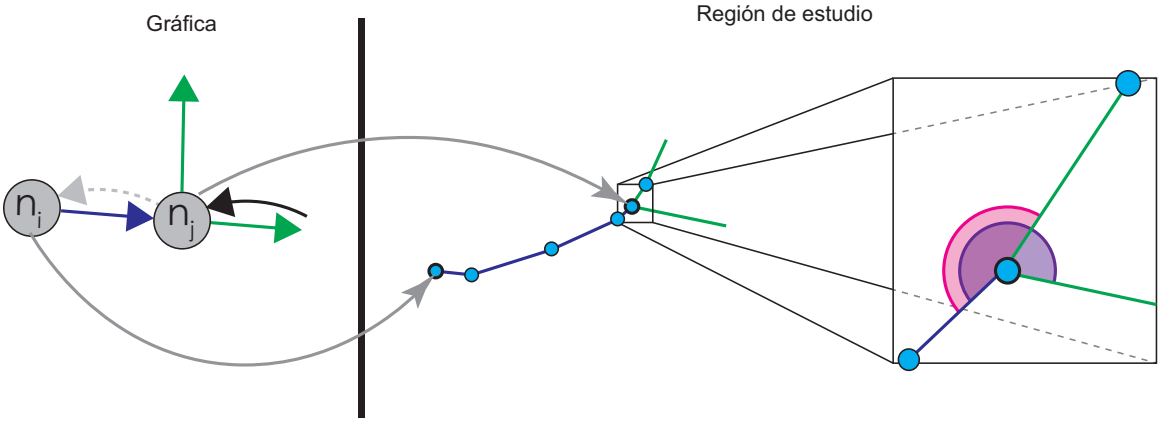


Figura A.6: Paso 3, del proceso principal

A continuación se va a mostrar con imágenes cómo se va desarrollando este algoritmo con este ejemplo.

Siguiendo el algoritmo, se debe empezar con n_1 y tomar su arista de contorno, para este caso es la arista $(\{n_1, n_2\}, C)$ (Figura A.9a).

Ahora se sigue con el nodo n_2 y luego con esta la tripleta (n_2, $(\{n_1, n_2\}, C)$ y las aristas que salen de n_2). Entonces se elige la arista de ángulo mínimo, en este caso $\{n_2, n_{11}\}$ (Figura A.11a).

El siguiente es el nodo n_{11}, y sigue obtener el conjunto de aristas que salen de este (Figura A.12a). En este caso, de las aristas salientes se descarta la arista $\{n_{11}, n_2\}$, ya que se llegó al nodo n_{11} usando la arista $\{n_2, n_{11}\}$, y no tiene sentido regresar. Luego hay que verificar los ángulos que se forman en la región geométrica asociada al nodo n_{11}, la arista $\{n_2, n_{11}\}$ y las aristas que salen de n_{11}. Entonces se elige la arista de ángulo mínimo, en este caso $\{n_{11}, n_{30}\}$ (Figura A.13a).

Se continúa con el nodo n_{30}, luego se toma la tripleta (n_{30}, $\{n_{11}, n_{30}\}$, las aristas que salen de n_{30}) para elegir la arista de ángulo mínimo, en este caso $\{n_{30}, n_1\}$ (Figura A.15a). Al llegar al nodo inicial (n_1), se concluye con la construcción de la región (Figura A.15c). Es importante recordar que las aristas de la región encontrada (en este caso $\{n_1, n_2\}$,$\{n_2, n_{11}\}$,$\{n_{11}, n_{30}\}$,$\{n_{30}, n_1\}$) fueron retirados de la gráfica.

Al haber concluido con una región se sigue con el nodo sucesor en numeración al 1 (que fue con el que se empezó la región anterior), es decir, el nodo 2 (n_2). De la misma forma que con el nodo 1, se toma la arista de contorno (etiquetada con la letra C) que sale del nodo 2 (Figura A.16a).

Ahora se sigue con el nodo n_3 y obtener el conjunto de aristas que salen de este (Figura A.17a). Se sigue con la tripleta (n_3, $(\{n_2, n_3\}, C)$, las aristas que salen de n_3). Entonces se elige la arista de ángulo mínimo, en este caso $\{n_3, n_{14}\}$ (Figura A.18a).

Sigue el nodo n_{14} y obtener el conjunto de aristas que salen de este (Figura A.19a), quitando la arista $\{n_{14}, n_3\}$ ya que se llegó al nodo n_4 usando la arista $\{n_3, n_{14}\}$. Y la

tripleta es (n_{14}, $\{n_3, n_{14}\}$, las aristas que salen de n_{14}). La arista de ángulo mínimo, en este caso es $\{n_{14}, n_{11}\}$ (Figura A.20a).

Se continúa con el nodo n_{11}, con la tripleta (n_{11}, $\{n_{14}, n_{11}\}$,las aristas que salen de n_{11}). Al final se elije $\{n_{11}, n_2\}$ (Figura A.22a). Como se llega al nodo n_2, que es con el que se empezó, entonces se concluye con la construcción de la región (Figura A.22c). Es importante recordar que las aristas de la región encontrada (en este caso $\{n_2, n_3\}$,$\{n_3, n_{14}\}$,$\{n_{14}, n_{11}\}$,$\{n_{11}, n_2\}$) fueron retiradas de la gráfica, y como el nodo 2 ya no tiene vértices, se quita de la gráfica.

Ahora se continua con el nodo 3 (n_3). Se sigue tomando la arista de contorno (Etiquetada con la letra C) que sale del nodo 3, y se continua de la misma forma hasta tener ocho regiones. Se llega a la gráfica que se ve en la figura A.7. Las siguientes regiones son diferentes a las anteriores, por lo que vale la pena ver con detalle como se hacen.

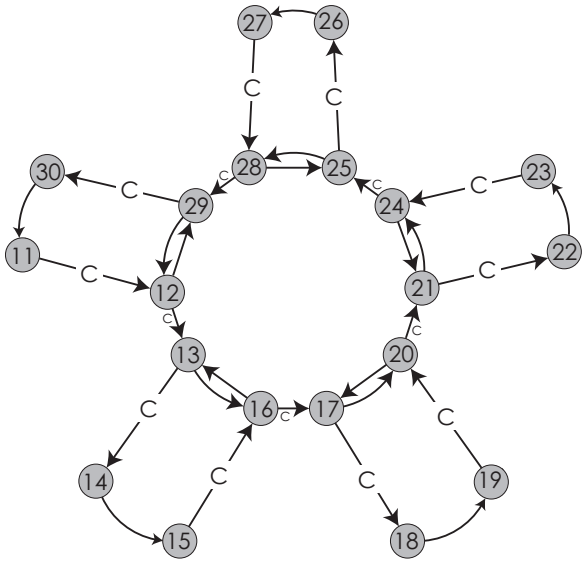


Figura A.7: Gráfica después de quitar las primeras 10 regiones.

Sigue el nodo n_{11}. Al igual que en las anteriores regiones se debe tomar la arista de contorno que salga de éste (o en su defecto, si no existiera, buscar la arista $\{u, v\}$ tal que no exista $\{v, u\}$), en este caso la arista $(\{n_{11}, n_{12}\}, C)$ (Figura fig:CGraph0). Esto nos lleva al nodo doce.

Sigue el nodo n_{12} y obtener el conjunto de aristas que salen de este (Figura A.24a). Posteriormente con la tripleta (n_{12}, $(\{n_{11}, n_{12}\}, C)$, las aristas que salen de n_{12}) se obtiene la arista asociada al ángulo mínimo, en este caso $\{n_{12}, n_{29}\}$ (Figura A.25a).

La arista lleva al nodo n_{29}, para posteriormente obtener el conjunto de aristas que salen de este (Figura A.26a), se descarta la arista $\{n_{29}, n_{12}\}$ ya que se llegó al nodo n_{29} usando la arista $\{n_{12}, n_{29}\}$; por lo que sólo nos queda la arista $\{n_{29}, n_{11}\}$ (La cuál se toma) (Figura A.27a).

Se continúa con el nodo n_{30}, se obtiene el conjunto de aristas que salen de este (Figura A.28a). En este caso la única arista que sale del nodo 30 es $\{n_{11}, n_2\}$ (Figura A.29a). Al llegar al nodo inicial (n_{11}), se concluye con la construcción de la región (Figura A.29c). Es importante recordar que las aristas de la región encontrada (en este caso $\{n_{11}, n_{12}\}$,$\{n_{12}, n_{29}\}$,$\{n_{29}, n_{30}\}$,$\{n_{30}, n_{11}\}$) fueron retirados de la gráfica, y

como el nodo 30 y 11 ya no tienen aristas, se quitan de la gráfica.

Al haber concluido con esta región, se sigue con el nodo sucesor en numeración al 11 (que fue con el que se empezó la región anterior), es decir, el nodo 12 (n_{12}). De la misma forma que con el nodo 11, se toma la arista de contorno (etiquetada con la letra C), que sale del nodo 12 (Figura A.30a).

Se debe seguir entonces con el nodo 13, y obtener el conjunto de aristas que salen de éste (Figura A.31a). Luego hay que verificar los ángulos que se forman en la región geométrica asociada al nodo n_{13}, la arista $(\{n_{12}, n_{13}\}, C)$ y las aristas que salen de n_{13}. Entonces se elige la arista de ángulo mínimo, en este caso $\{n_{13}, n_{16}\}$ (Figuras A.31 y A.32).

Esta arista lleva al nodo n_{16}, ahora se debe usar la tripleta (n_{16}, $\{n_{13}, n_{16}\}$, $(\{n_{16}, n_{17}\}, C)$) (no se puede tomar en cuenta $\{n_{16}, n_{13}\}$), se toma claramente la arista $\{n_{16}, n_{17}\}$ (Figura A.32a). Se continúa de manera análoga, y al final se llegan a las siguientes aristas: $(\{n_{12}, n_{13}\}, C)$, $\{n_{13}, n_{16}\}$, $(\{n_{16}, n_{17}\}, C)$, $\{n_{17}, n_{20}\}$, $(\{n_{20}, n_{21}\}, C)$, $\{n_{21}, n_{24}\}$, $(\{n_{24}, n_{25}\}, C)$, $\{n_{25}, n_{28}\}$, $(\{n_{28}, n_{29}\}, C)$, $\{n_{29}, n_{12}\}$. Éstas forman la nueva región, se quitan estas aristas de la gráfica, los nodo 12 y 29 se quedan sin aristas, y por esto se quitan de la gráfica (Figura A.33).

Ahora la gráfica tiene nodos en los cuales sólo hay una arista de salida en cada uno, por lo tanto el algoritmo consiste simplemente en recorrer las aristas hasta cerrar la región (Figura A.8).

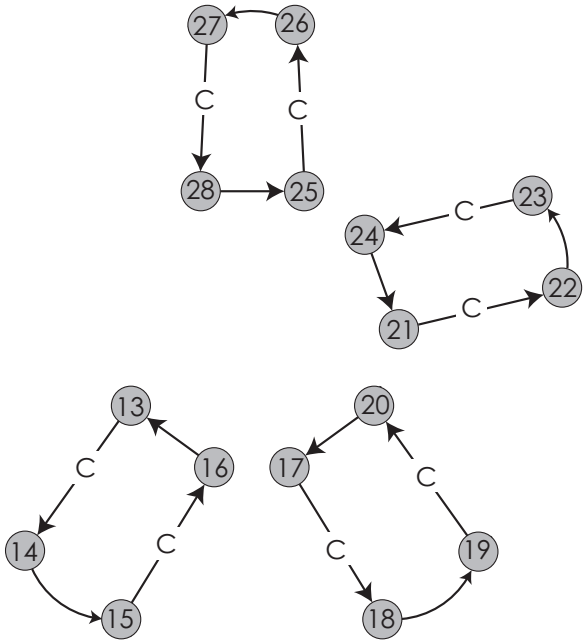
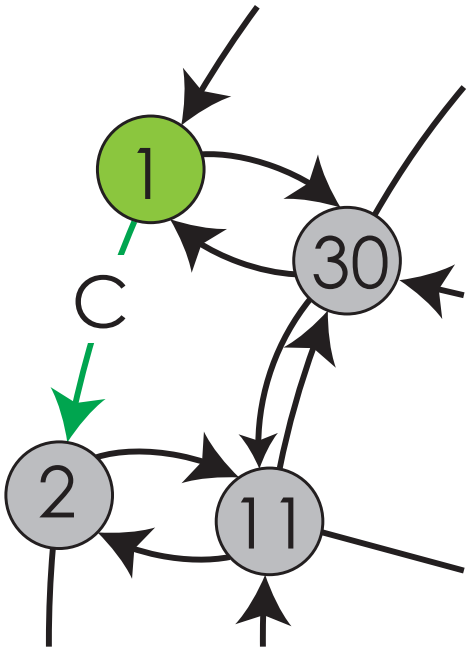
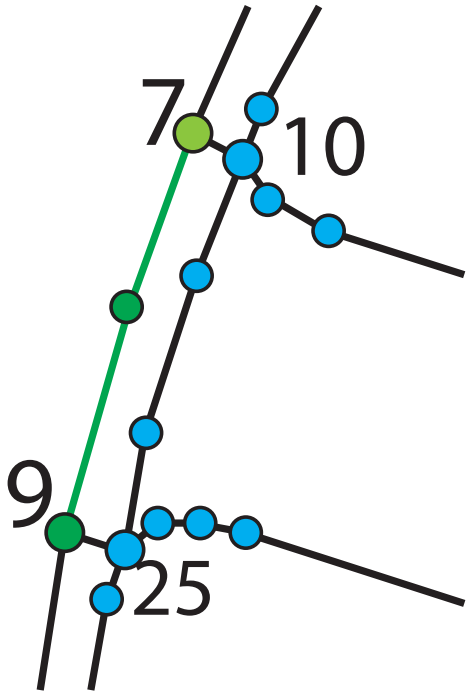


Figura A.8: Gráfica

Se continúa con el nodo 13 (pues la región anterior empezó con el nodo 12), en éste la arista que sale es $(\{n_{13}, n_{14}\}, C)$, y a su vez salen las aristas $\{n_{14}, n_{15}\}$, $(\{n_{15}, n_{16}\}, C)$, $\{n_{16}, n_{13}\}$, con lo que se cierra la región y se sigue análogamente con las siguientes tres regiones.

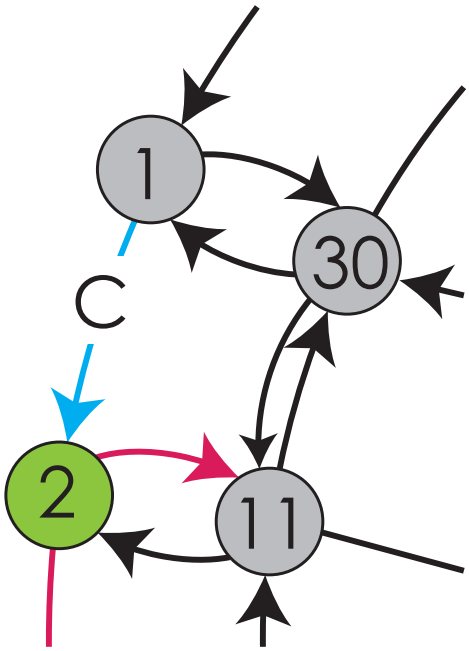


(a) Nodo n_1 y la arista seleccionada para avanzar.

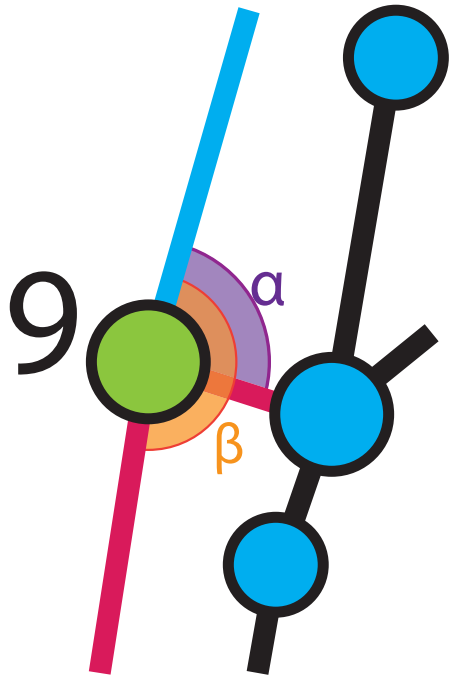


(b) Vértice asociado al nodo n_1 y subconjunto poligonal asociado a la arista seleccionada.

Figura A.9

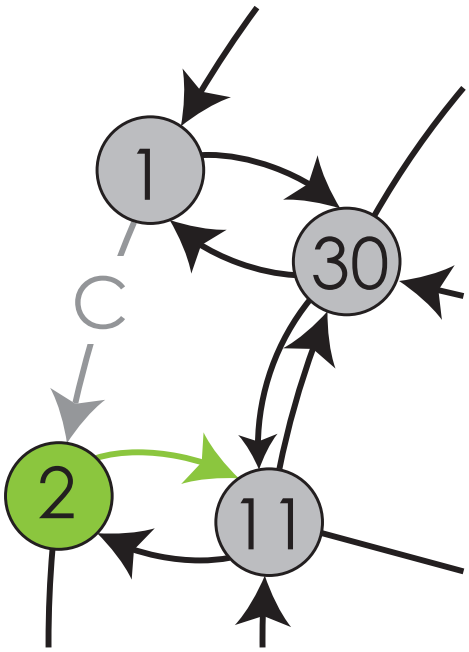


(a) Nodo n_2, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

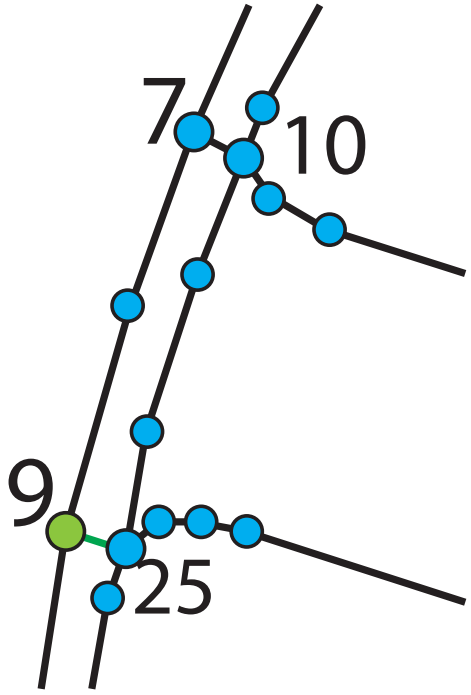


(b) Vértice asociado al nodo n_2 y los ángulos asociados a las aristas salientes.

Figura A.10

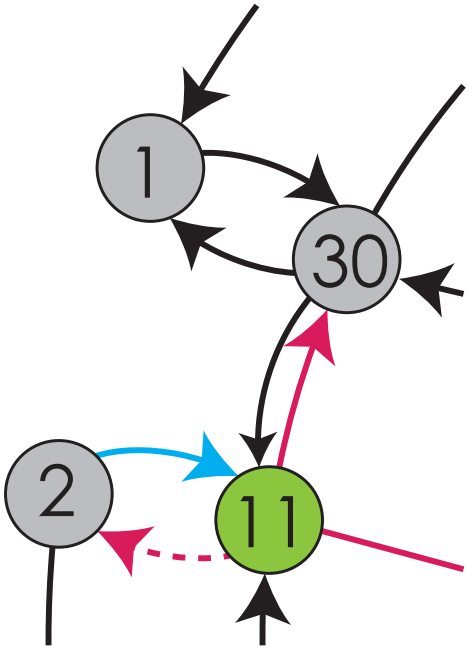


(a) Nodo n_2 y la arista seleccionada para avanzar, en gris la arista que ya forma parte de la región pero ya no parte de la gráfica.

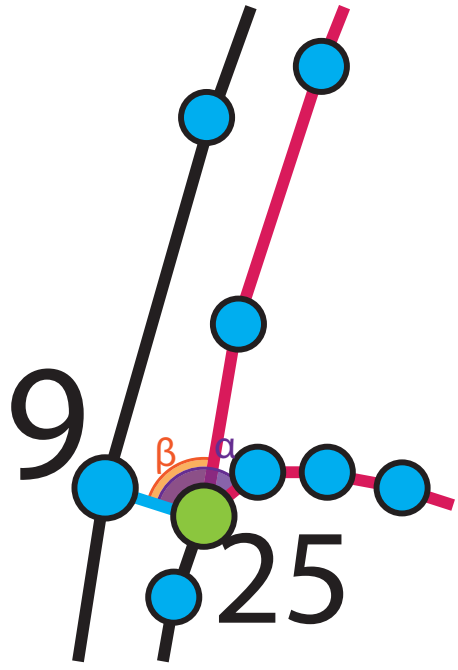


(b) Vértice asociado al nodo n_2 y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.11

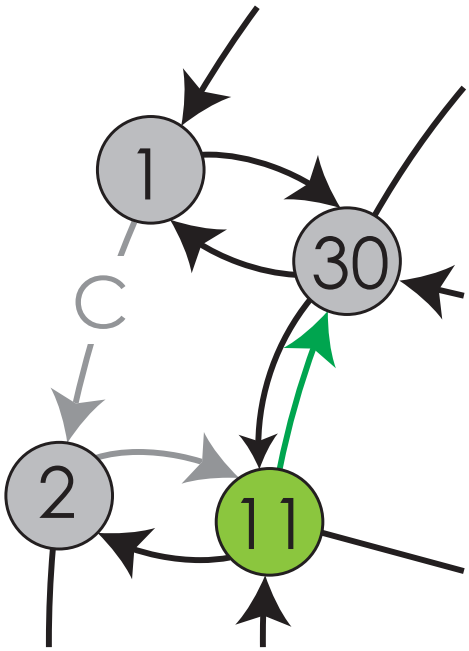


(a) Nodo n_{11}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

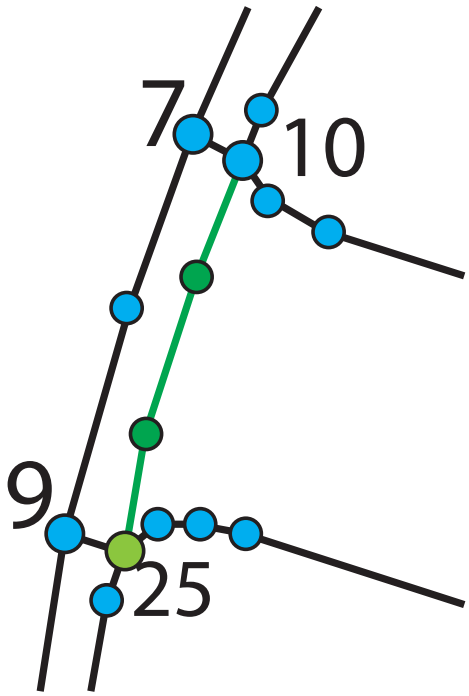


(b) Vértice asociado al nodo n_{11} y los ángulos asociados a las aristas salientes, la arista punteada se descarta al ser contraria a la azul.

Figura A.12

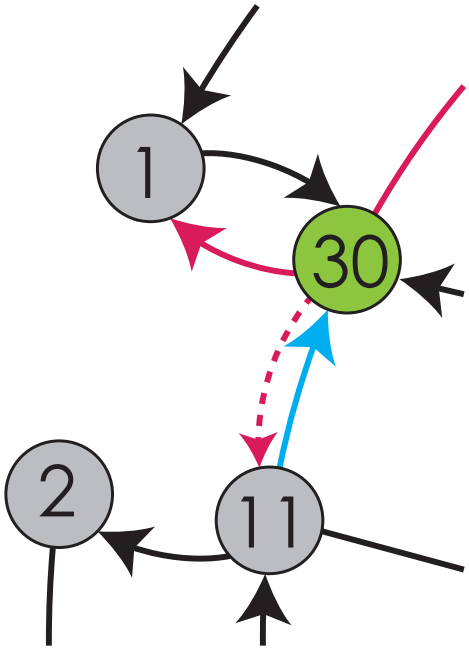


(a) Nodo n_{11} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región pero ya no de la gráfica.

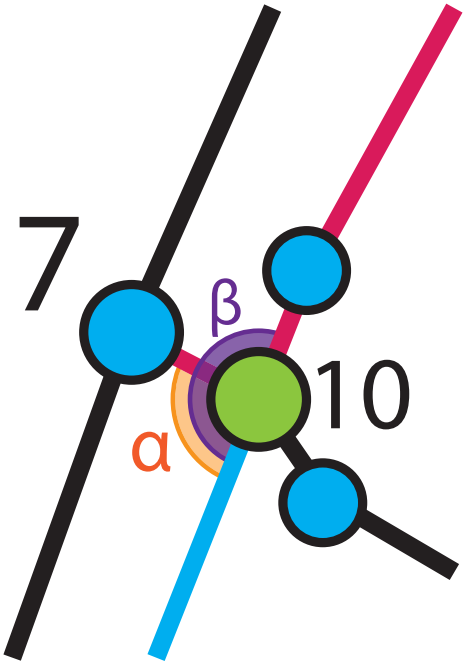


(b) Vértice asociado al nodo n_2 y subconjunto poligonal asociado a la arista seleccionada.

Figura A.13

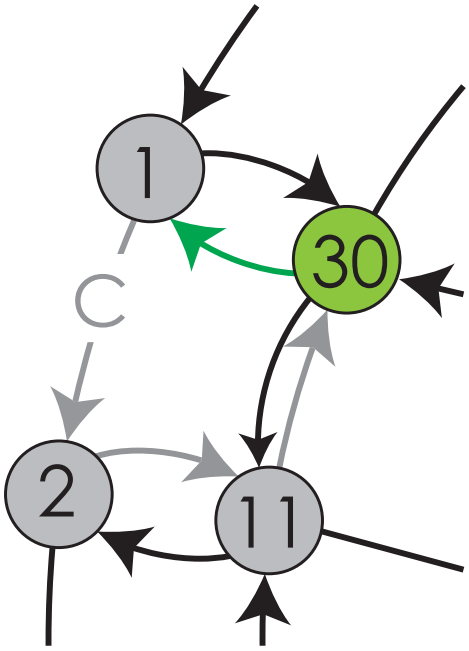


(a) Nodo n_{30}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

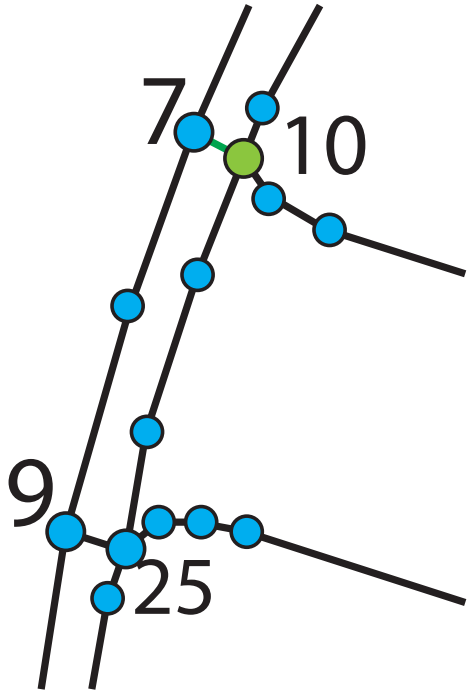


(b) Vértice asociado al nodo n_{30} y los ángulos asociados a las aristas salientes, la arista punteada se descarta al ser contraria a la azul.

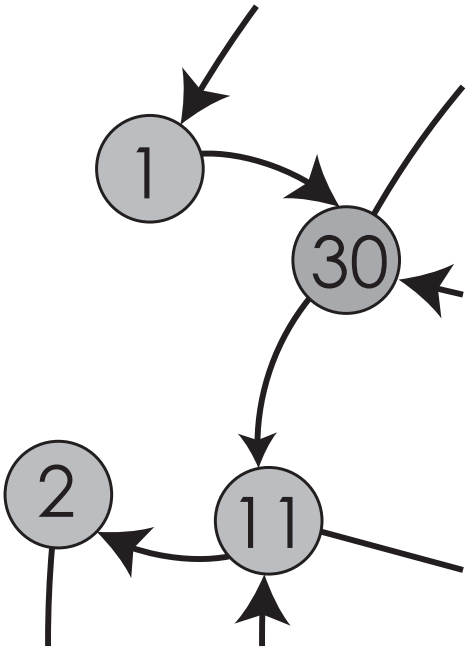
Figura A.14



(a) Nodo n_{30} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región pero ya no de la gráfica.

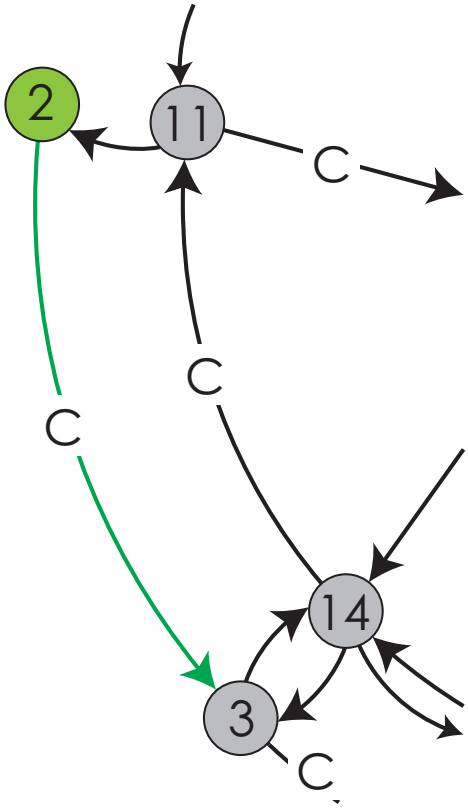


(b) Vértice asociado al nodo n_{30} y subconjunto poligonal asociado a la arista seleccionada.

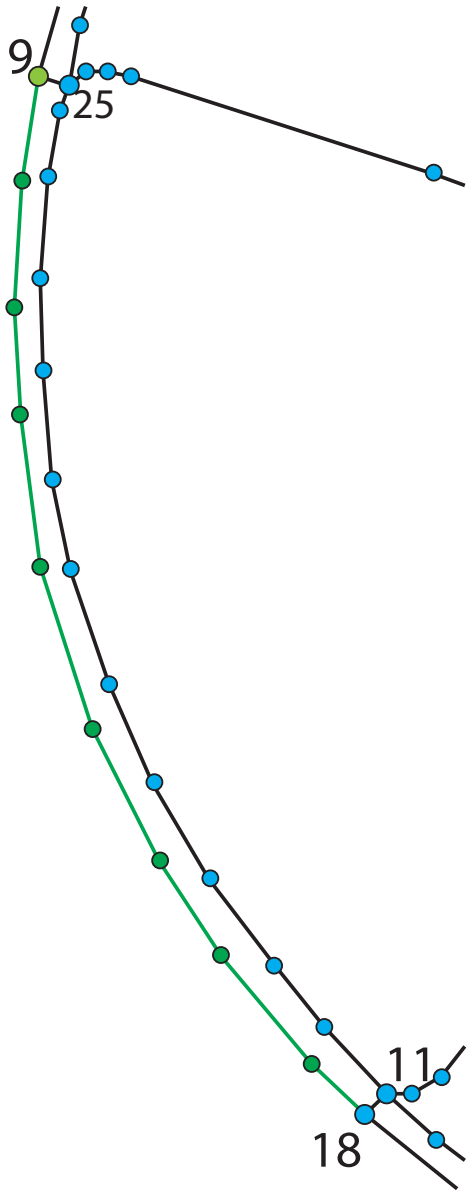


(c) Al final del proceso, la gráfica queda sin las aristas de la región encontrada.

Figura A.15

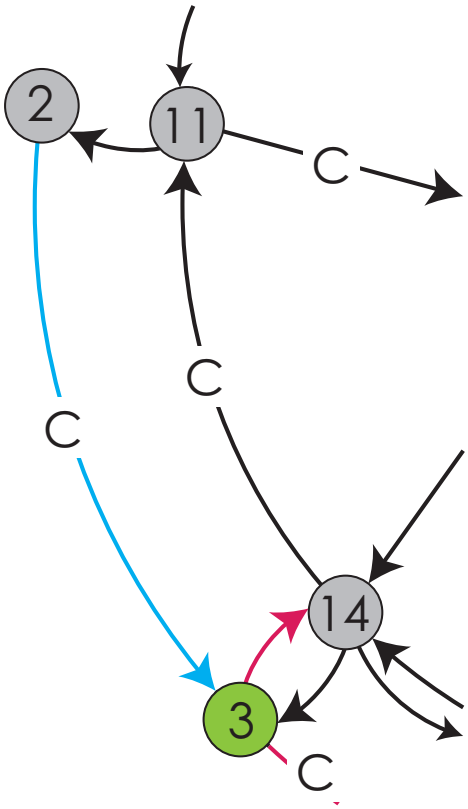


(a) Nodo n_2 y la arista seleccionada para avanzar.

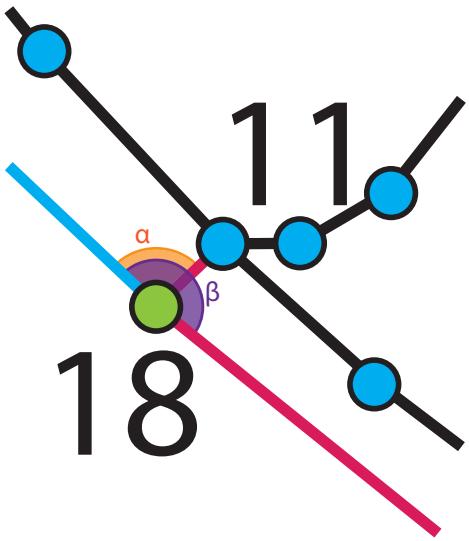


(b) Vértice asociado al nodo n_2 y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.16

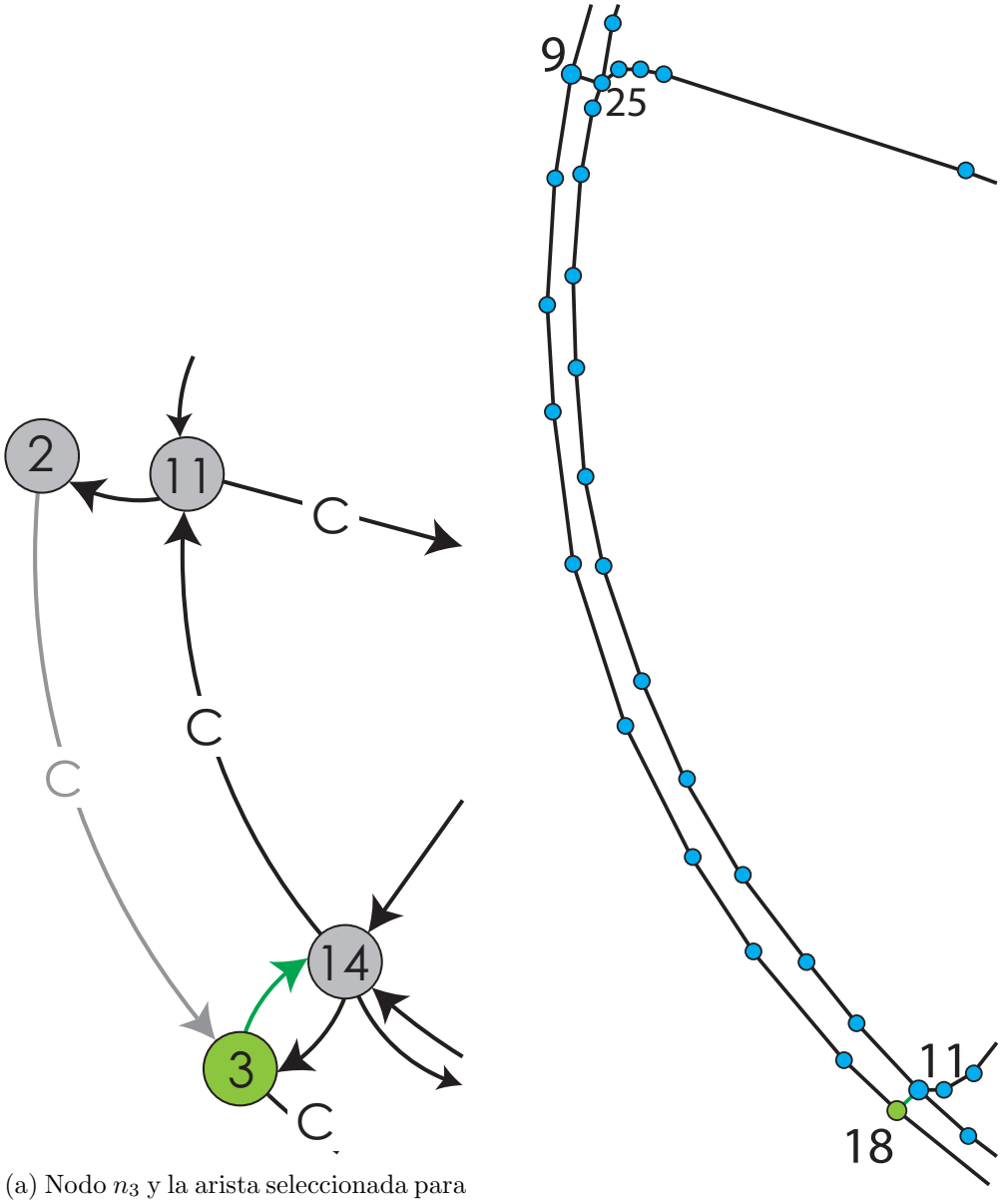


(a) Nodo n_3, en azul la arista con la que se llegó al nodo, y en purpura las aristas salientes del nodo.



(b) Vértice asociado al nodo n_3 y los ángulos asociados a las aristas salientes.

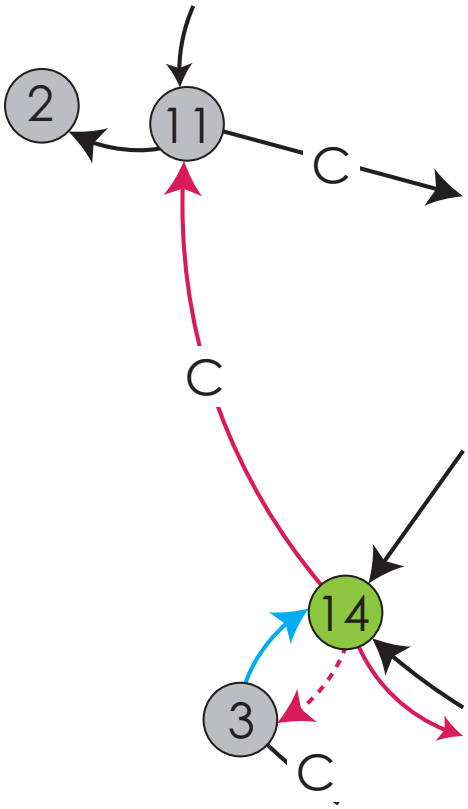
Figura A.17



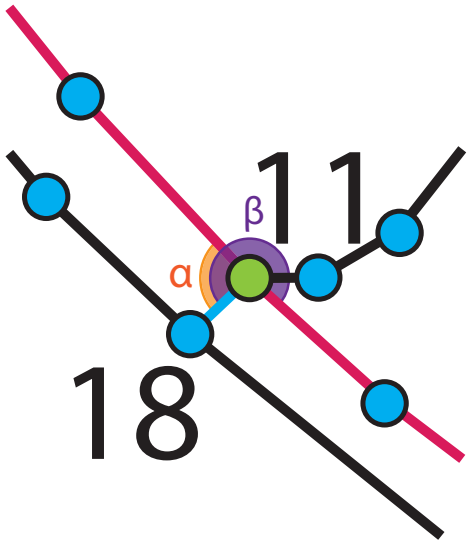
(a) Nodo n_3 y la arista seleccionada para avanzar, en gris la arista que ya forma parte de la región, pero ya no parte de la gráfica.

(b) Vértice asociado al nodo n_3 y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.18

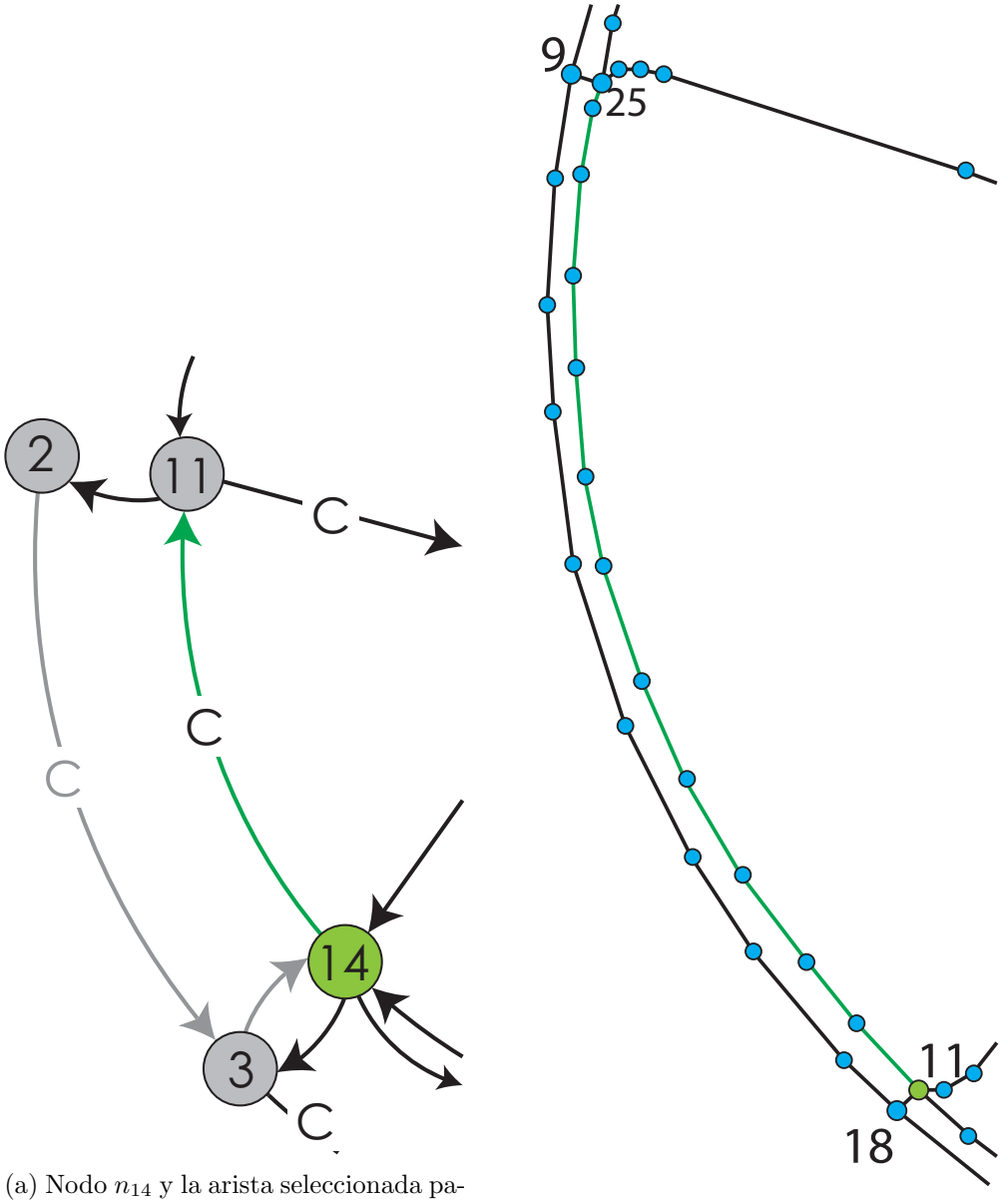


(a) Nodo n_{14}, en azul la arista con la que se llegó al nodo, y en purpura las aristas salientes del nodo.



(b) Vértice asociado al nodo n_{14} y los ángulos asociados a las aristas salientes. La arista punteada se descarta al ser contraria a la azul.

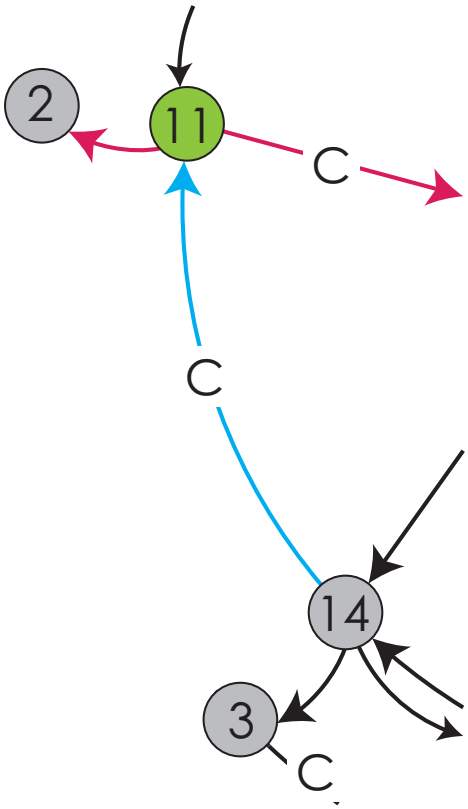
Figura A.19



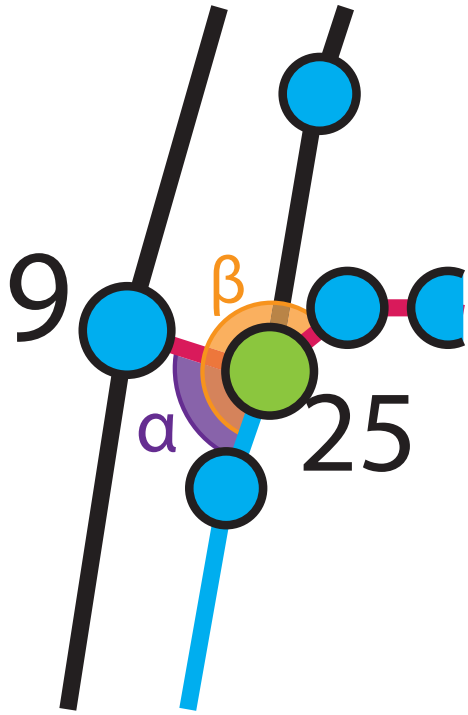
(a) Nodo n_{14} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región, pero ya no de la gráfica.

(b) Vértice asociado al nodo n_{14} y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.20

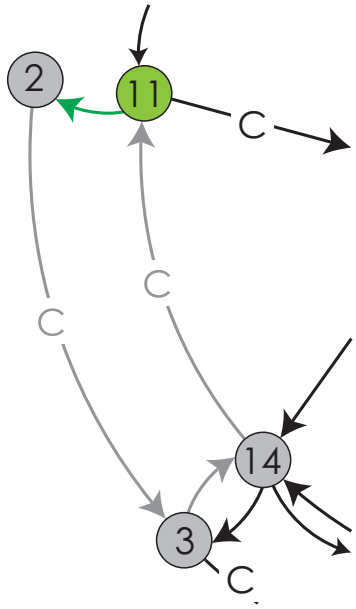


(a) Nodo n_{11}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

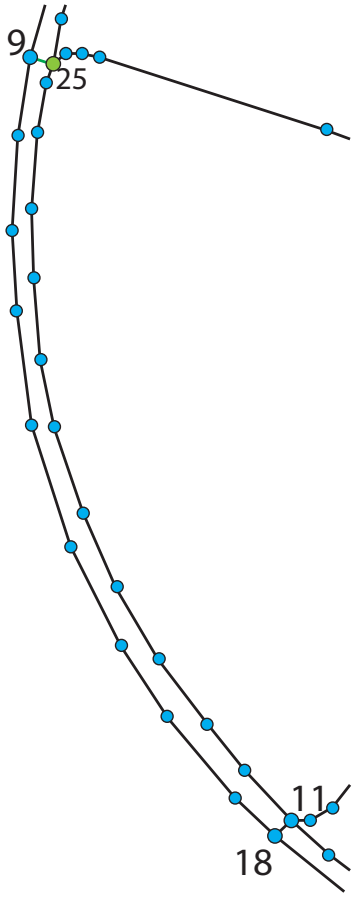


(b) Vértice asociado al nodo n_{11} y los ángulos asociados a las aristas salientes, la arista punteada se descarta al ser contraria a la azul.

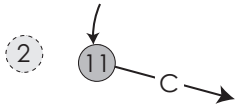
Figura A.21

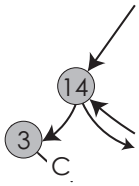


(a) Nodo n_{11} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región pero ya no de la gráfica.



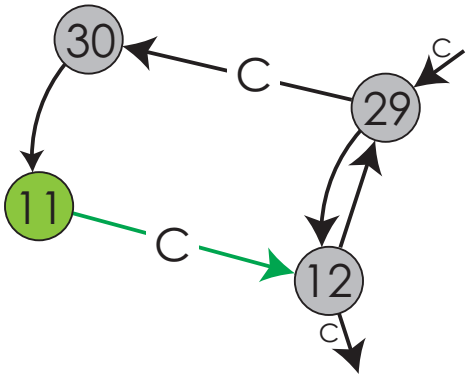
(b) Vértice asociado al nodo n_{11} y sunconjunto poligonal asociado a la arista seleccionada.



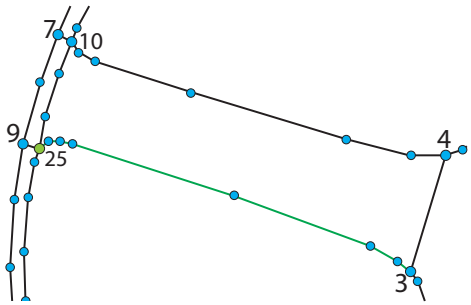


(c) Al final del proceso, la gráfica queda sin las aristas de la región encontrada y quitando el nodo sin arista.

Figura A.22

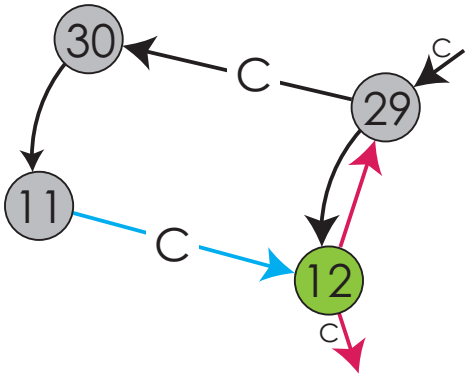


(a) Nodo n_{11} y la arista seleccionada para avanzar.

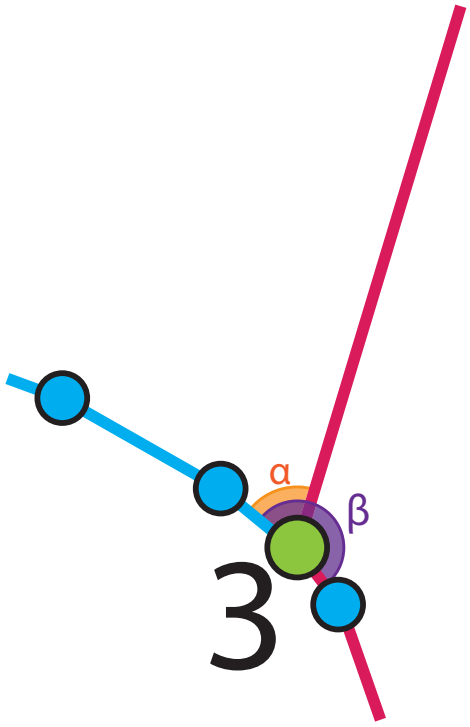


(b) Vértice asociado al nodo n_{11} y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.23

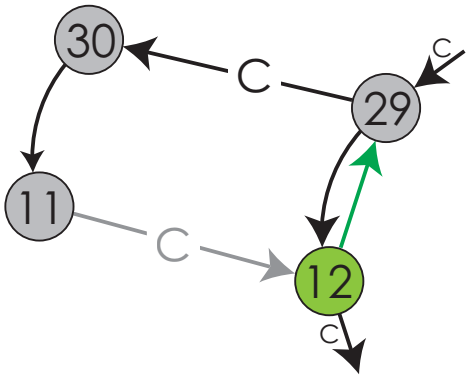


(a) Nodo n_{12}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

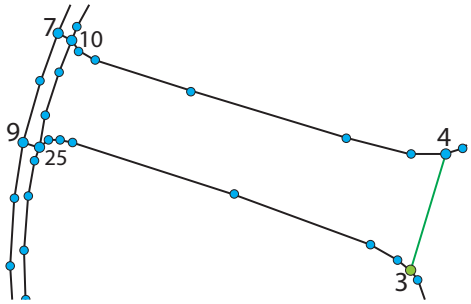


(b) Vértice asociado al nodo n_{12} y los ángulos asociados a las aristas salientes.

Figura A.24

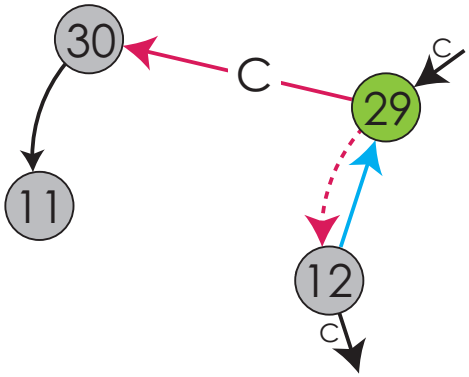


(a) Nodo n_{12} y la arista seleccionada para avanzar, en gris la arista que ya forma parte de la región, pero ya no parte de la gráfica.

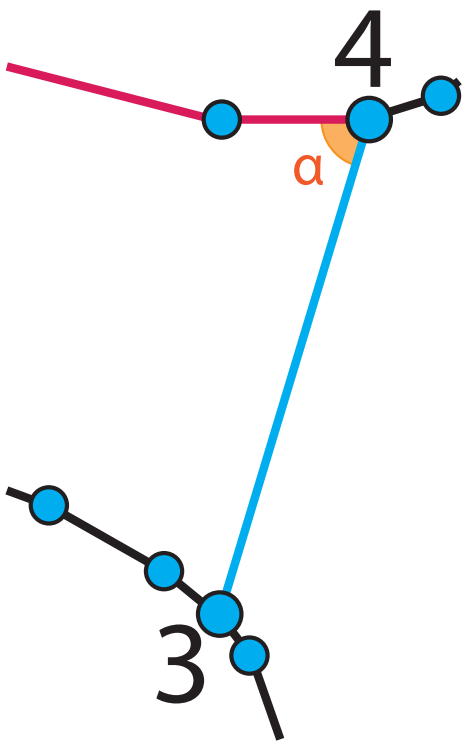


(b) Vértice asociado al nodo n_{12} y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.25

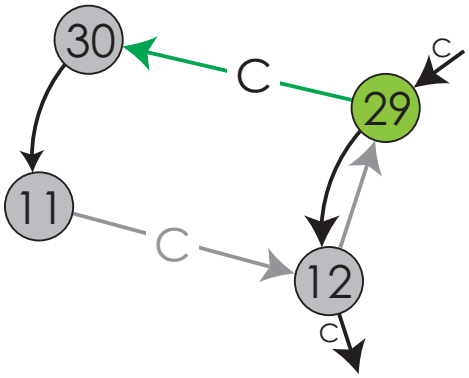


(a) Nodo n_{29}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

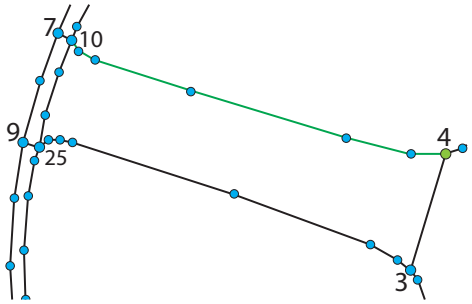


(b) Vértice asociado al nodo n_{29} y los ángulos asociados a las aristas salientes, la arista punteada se descarta al ser contraria a la azul.

Figura A.26

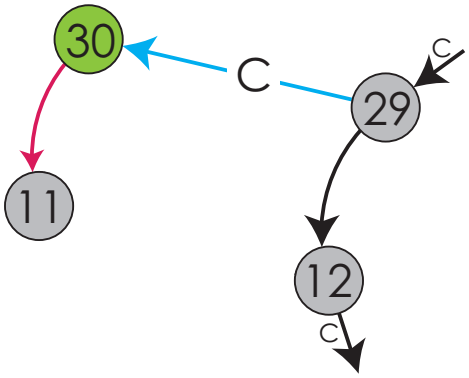


(a) Nodo n_{29} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región, pero ya no de la gráfica.

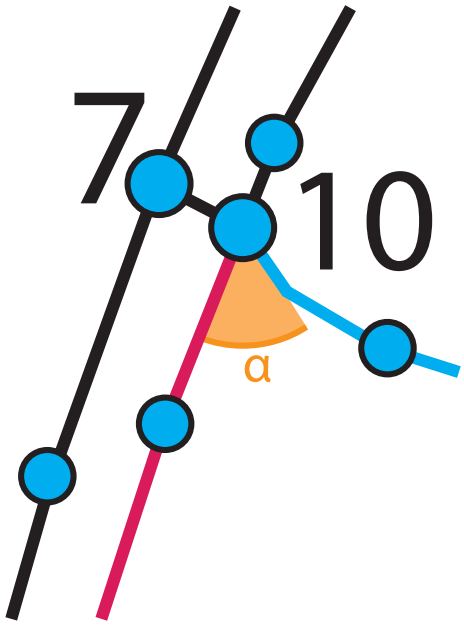


(b) Vértice asociado al nodo n_{29} y sunconjunto poligonal asociado a la arista seleccionada.

Figura A.27

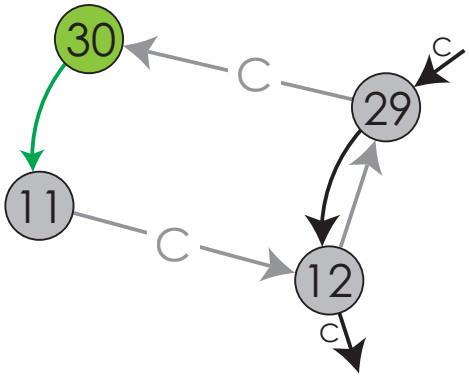


(a) Nodo n_{11}, en azul la arista con la que se llegó al nodo y en purpura las aristas salientes del nodo.

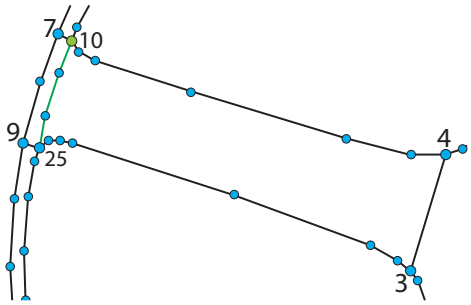


(b) Vértice asociado al nodo n_{11} y los ángulos asociados a las aristas salientes, la arista punteada se descarta al ser contraria a la azul.

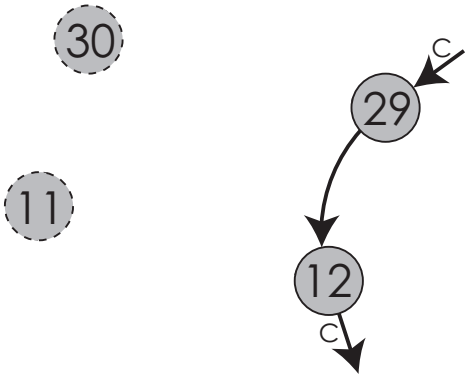
Figura A.28



(a) Nodo n_{11} y la arista seleccionada para avanzar, en gris las aristas que ya forman parte de la región, pero ya no de la gráfica.

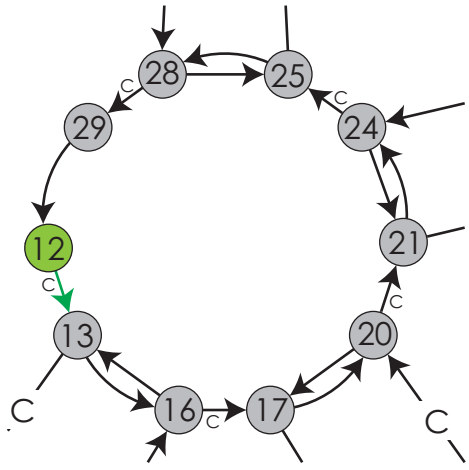


(b) Vértice asociado al nodo n_{11} y subconjunto poligonal asociado a la arista seleccionada.

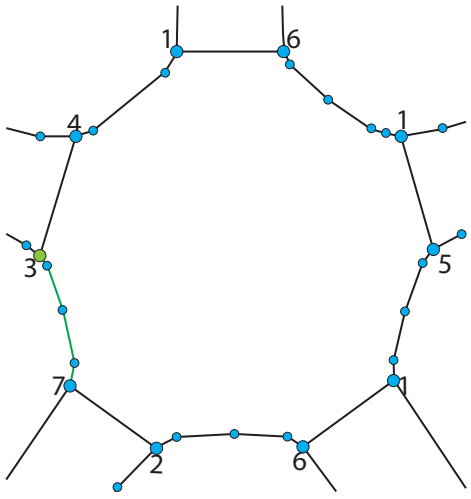


(c) Al final del proceso, la gráfica queda sin las aristas de la región encontrada y quitando los nodos sin aristas

Figura A.29

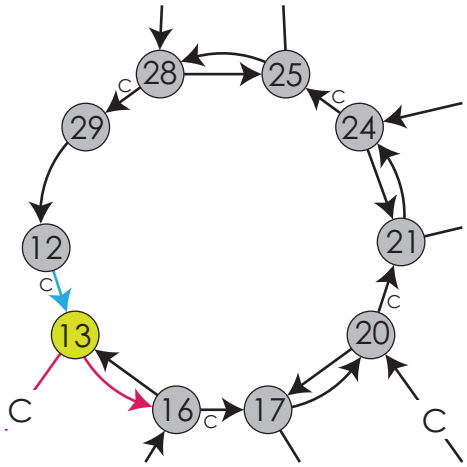


(a) Nodo n_{12} y la arista seleccionada para avanzar.

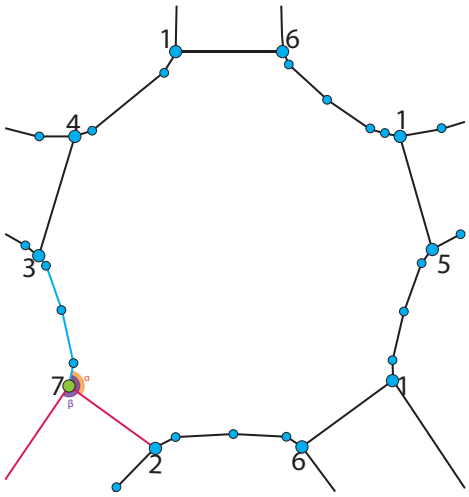


(b) Vértice asociado al nodo n_{12} y subconjunto poligonal asociado a la arista seleccionada.

Figura A.30

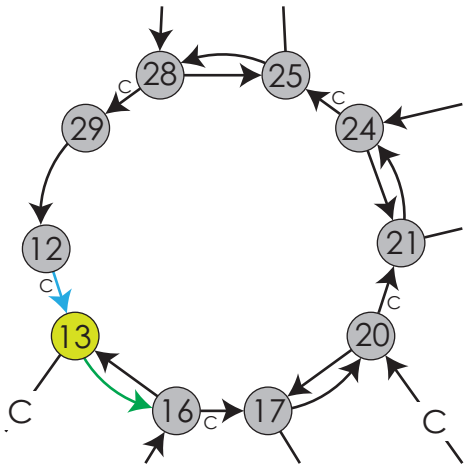


(a) Nodo n_{13} y las aristas posibles de avance.

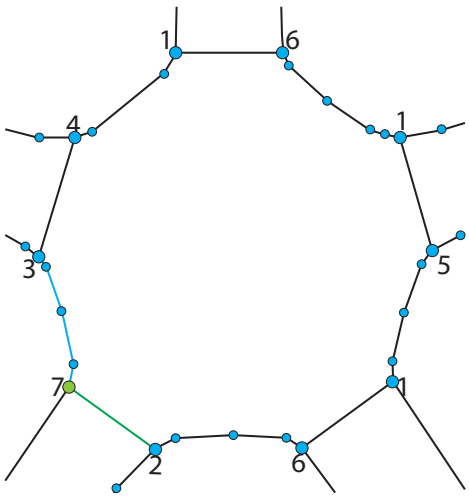


(b) Vértice asociado al nodo n_{13} y subconjunto poligonal asociado a las aristas que se analizan.

Figura A.31

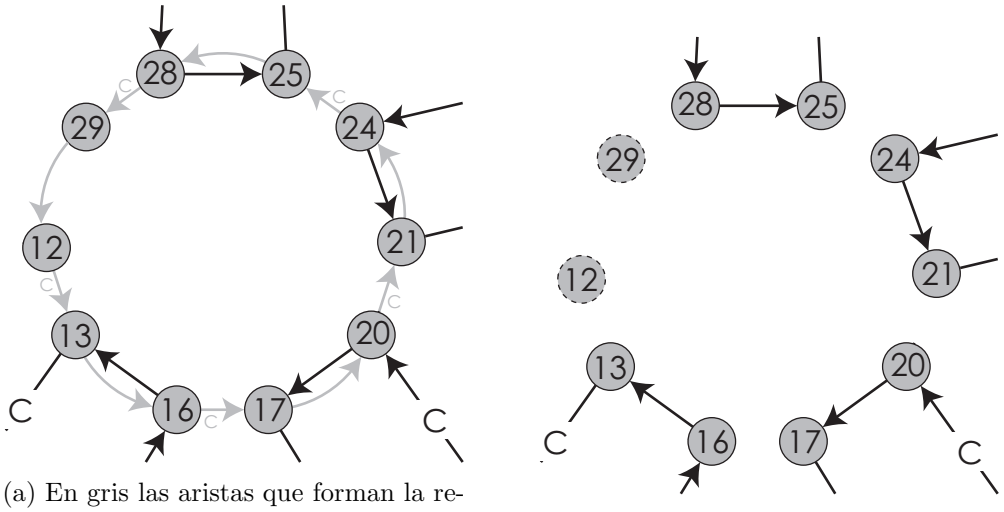


(a) Nodo n_{13} y la arista seleccionada para avanzar.



(b) Vértice asociado al nodo n_{13} y subconjunto poligonal asociado a la arista seleccionada.

Figura A.32



(a) En gris las aristas que forman la región.

(b) Gráfica sin aristas y sin los nodos

Figura A.33

A.3. Asignación de fronteras

Con lo anterior se termina el reconocimiento de regiones y se obtienen 16 subregiones en total, a cada subregión se le debe asignar las cuatro fronteras asociadas al cuadrado unitario. En este caso se asignaron las fronteras como se ve en la figura A.34, y las dimensiones de cada malla, como se indica en la lista A.1.

1. m=30, n=3	5. m=30, n=3	9. m=30, n=3	13. m=25, n=10
2. m=10, n=3	6. m=10, n=3	10. m=10, n=3	14. m=25, n=10
3. m=30, n=3	7. m=30, n=3	11. m=25, n=10	15. m=25, n=10
4. m=10, n=3	8. m=10, n=3	12. m=25, n=35	16. m=25, n=10

Lista A.1: Dimensiones de las subregiones, m corresponde a las fronteras impares, n a las pares

A.4. Generación de las mallas

Una vez asignadas las fronteras y las dimensiones, se debe seguir lo que indica el algoritmo 1.3.1 para generar la malla inicial de cada subregión, la idea es utilizar interpolación transfinita, por lo que es necesario primero generar los puntos de la malla en las cuatro fronteras. Para esto el algoritmo empieza ordenando por longitud las fronteras de cada subregión, lo cuál se aprecia en la lista A.2. Luego se debe asignar en cada frontera la cantidad de puntos que se pidieron, pero antes se debe verificar si la frontera tiene intersección con otra frontera anterior en la lista A.2, si es así se toman sus partes, y si no es así, se llena interpolando puntos de manera lineal, hasta terminar con todas las fronteras de la lista. A modo de ilustración se mostrará el caso de la frontera 54 en la lista, por ser un caso no tan sencillo.

1. (2,ω_2)	14. (4,ω_7)	27. (1,ω_2)	40. (1,ω_6)	53. (1,ω_{13})
2. (4,ω_3)	15. (2,ω_{10})	28. (2,ω_{14})	41. (3,ω_{12})	54. (3,ω_{13})
3. (4,ω_4)	16. (4,ω_1)	29. (3,ω_4)	42. (1,ω_{12})	55. (3,ω_{10})
4. (2,ω_3)	17. (4,ω_8)	30. (4,ω_{13})	43. (2,ω_{12})	56. (3,ω_6)
5. (2,ω_1)	18. (2,ω_7)	31. (1,ω_4)	44. (4,ω_{12})	57. (3,ω_8)
6. (4,ω_2)	19. (2,ω_8)	32. (3,ω_{10})	45. (3,ω_{14})	58. (3,ω_3)
7. (2,ω_4)	20. (4,ω_9)	33. (4,ω_{11})	46. (1,ω_{11})	59. (3,ω_1)
8. (4,ω_5)	21. (4,ω_{14})	34. (1,ω_{10})	47. (1,ω_{14})	60. (1,ω_5)
9. (2,ω_5)	22. (2,ω_{11})	35. (2,ω_{16})	48. (3,ω_{15})	61. (1,ω_9)
10. (4,ω_6)	23. (4,ω_{15})	36. (3,ω_8)	49. (3,ω_{11})	62. (1,ω_3)
11. (2,ω_9)	24. (3,ω_2)	37. (1,ω_8)	50. (1,ω_{15})	63. (1,ω_7)
12. (4,ω_{10})	25. (2,ω_{13})	38. (2,ω_{15})	51. (3,ω_{16})	64. (1,ω_1)
13. (2,ω_6)	26. (4,ω_{16})	39. (3,ω_6)	52. (1,ω_{16})	

Lista A.2: Orden de las fronteras de cada subregión. El formato es (F, ω_i), en el que F es el número de frontera, y ω_i la subregión.

En el caso de la frontera 1 de la subregión ω_{12} se le pide tener 25 puntos (lista A.1), es el elemento 42 de la lista A.2, e intersecta con la frontera 4 de la subregión ω_{13}, a ésta ya le fueron asignados 10 puntos anteriormente, pues es el elemento 30 de la lista A.2. Además, la intersección está en medio del recorrido de la primer frontera, por lo que se parte en tres: una sin puntos asignados, otra corresponde a la intersección con los puntos ya asignados, y una más sin puntos asignados. En cada parte sin puntos asignados se asignan los puntos interpolando de manera lineal y finalmente se juntan evitando duplicar los puntos en común entre los interpolados (es decir, los extremos de los puntos asignados en la frontera 4 de la subregión ω_{13}). La cantidad de puntos que se le asigna a cada parte depende de su longitud, garantizando que haya al menos dos puntos por parte. Al juntar todos los puntos obtenemos la frontera completa, y se continúa con la siguiente frontera. Se puede apreciar de manera visual en la figura A.35.

Con esto se tienen los puntos de las fronteras ya asignados, y lo que sigue es utilizar el método de interpolación transfinita para generar la malla inicial de todas las subregiones (Figura A.36a) y después con algún método de suavizamiento se obtiene una malla suave de la región (Figura A.36b).

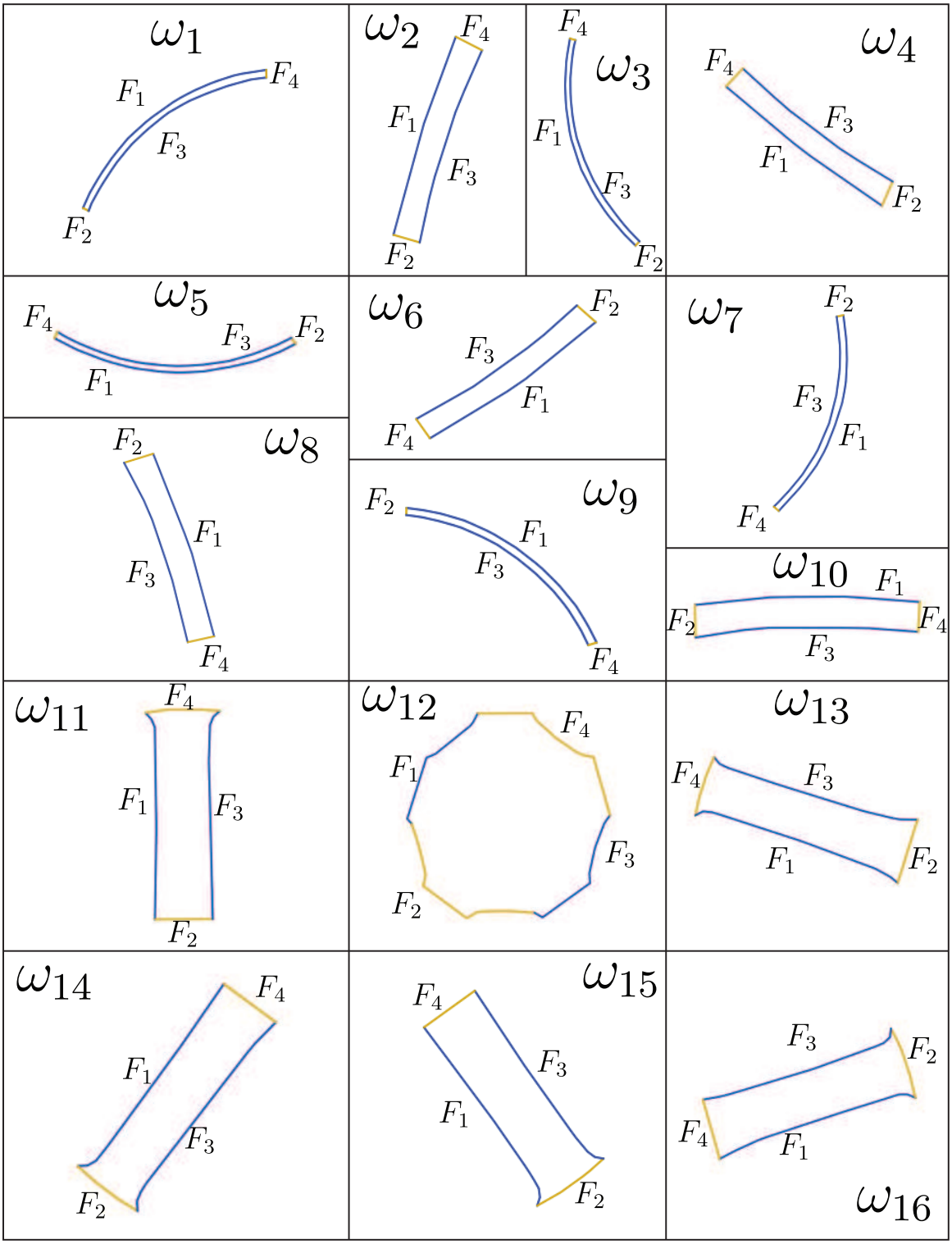


Figura A.34: Subregiones y sus fronteras numeradas

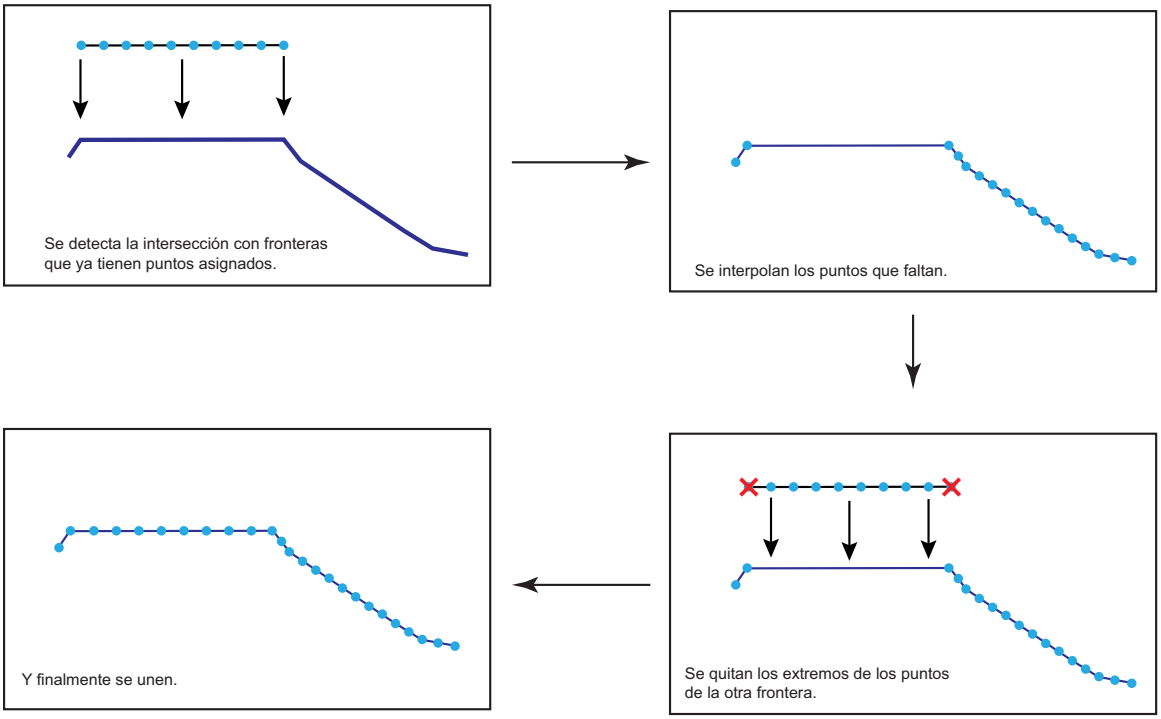
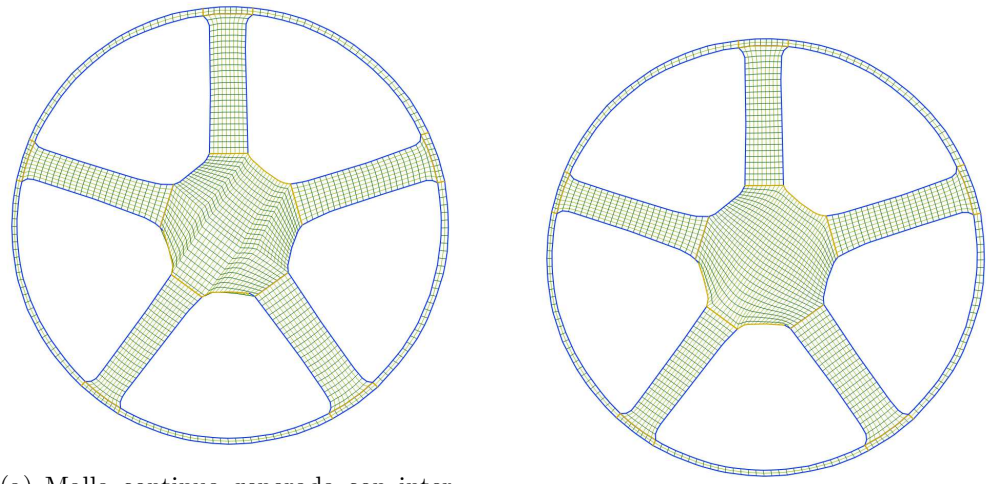


Figura A.35: Distribución de puntos a los largo de la frontera 1 de la subregión ω_{12}



(a) Malla continua generada con interpolación transfinita

(b) Malla suave

Figura A.36

Bibliografía

[1] P. Barrera and J. G. Tinoco. Area control in generating smooth and vonces grids over general plane regions. *Journal of Computational and Applied Mathematics*, 1999.

[2] P. Barrera and J.G. Tinoco. Smooth and convex grid generation over general plane regions. *Mathematics and Computers in Simulation*, 1998.

[3] P. Barrera-Sánchez, F.J. Domínguez-Mota, G.F. González-Flores, and J. G. Tinoco. Generating quality structured convex grids on irregular regions. *Electronic Transactions on Numerical Analysis*, 34:76–89, 2009.

[4] Gustavo Adolfo García Cano. Un sistema para la generación numérica de mallas estructuradas en regiones con agujeros. Tesis de licenciatura, UNAM, 2011.

[5] S. A. Ivanenko and A. A. Charakhch'yan. Curvilineal grids of convex quadrilaterals. *U.S.S.R. Comput. Maths. Phys.*, 28(2), 1998.