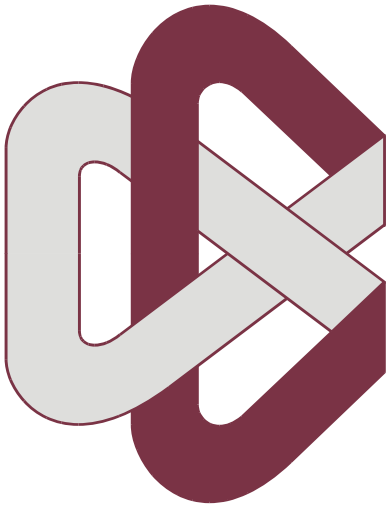


CIMAT

Método alternante de Schwarz

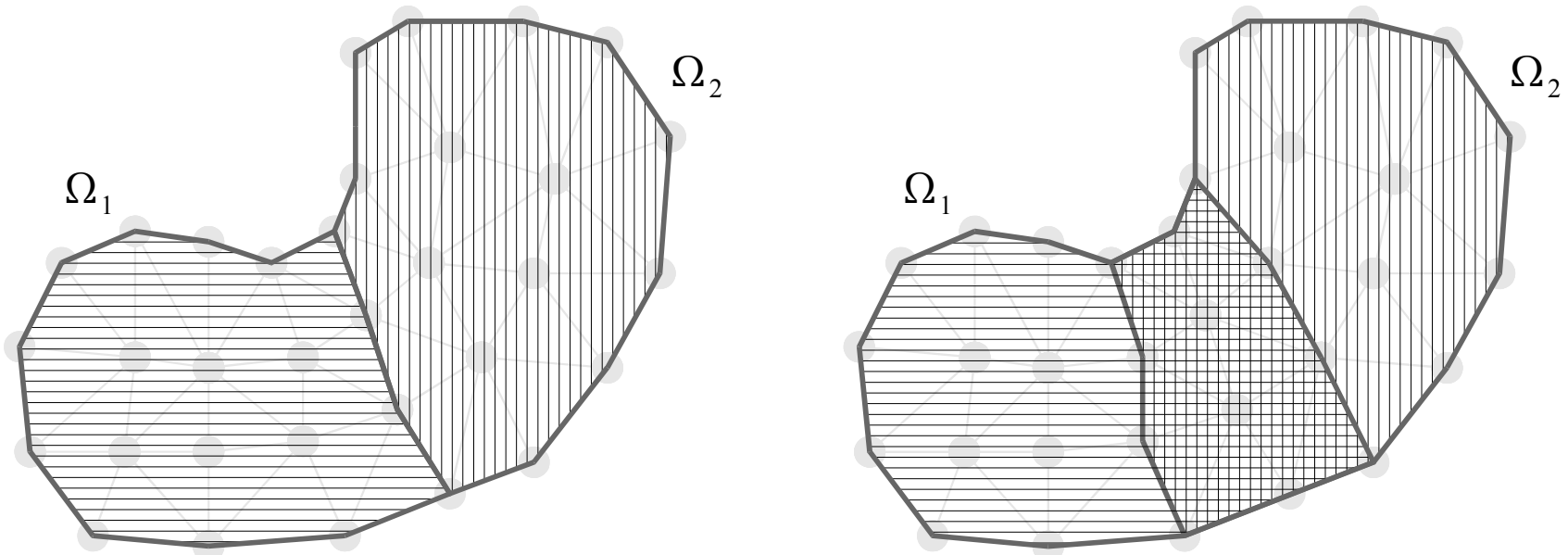
MSc Miguel Vargas-Félix
miguelvargas@cimat.mx
http://www.cimat.mx/~miguelvargas

Solución de ecuaciones diferenciales con descomposición de dominios

Al discretizar en millones de elementos, la cantidad de información que hace que el cálculo de la solución requiera tiempos de procesamiento y/o cantidades de memoria tales que no es posible resolver el problema utilizando una sola computadora en un tiempo razonable.

Se divide el dominio en particiones, para resolver cada una independientemente y después combinar las soluciones locales de forma iterativa.

Hay dos formas de trabajar la descomposición de dominio, con particiones con traslape y sin traslape.

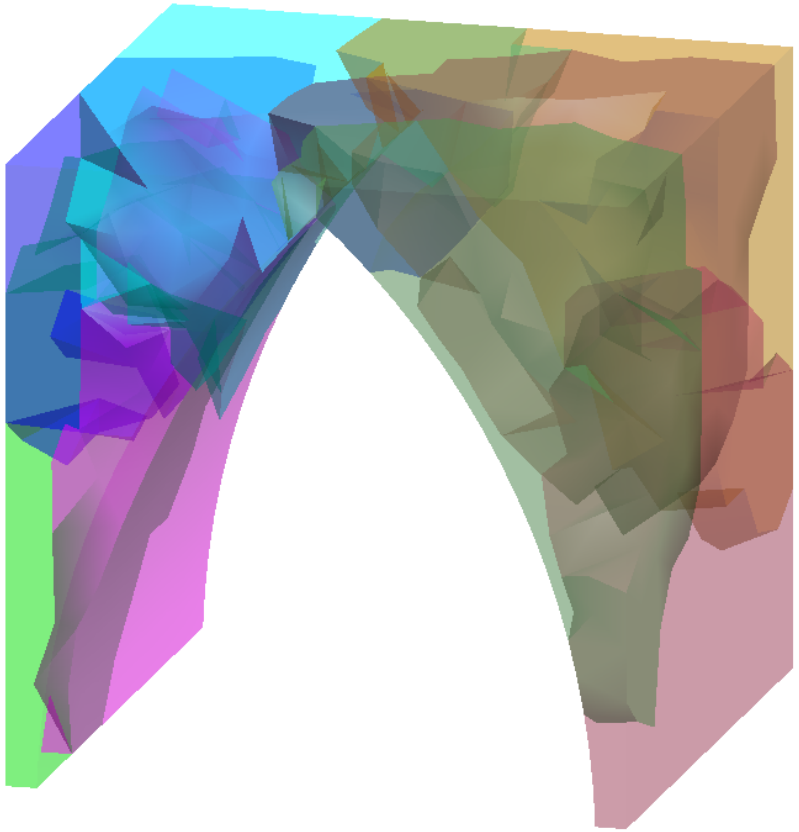


Descomposición en dominios sin traslape (izquierda) y con traslape (derecha).

Dividir un dominio en P particiones equivale entonces a separar en P bloques la matriz $\boldsymbol{A}$ que representa las relaciones entre nodos. Cada dominio entonces estará representado por una matriz $\boldsymbol{A}^{(p)}$, con $p = 1 \ldots P$.

El siguiente paso es resolver el sistema de ecuaciones con la matriz $\boldsymbol{A}^{(p)}$ de cada partición de foma independiente, utilizando algún método convencional para resolver sistemas de ecuaciones.

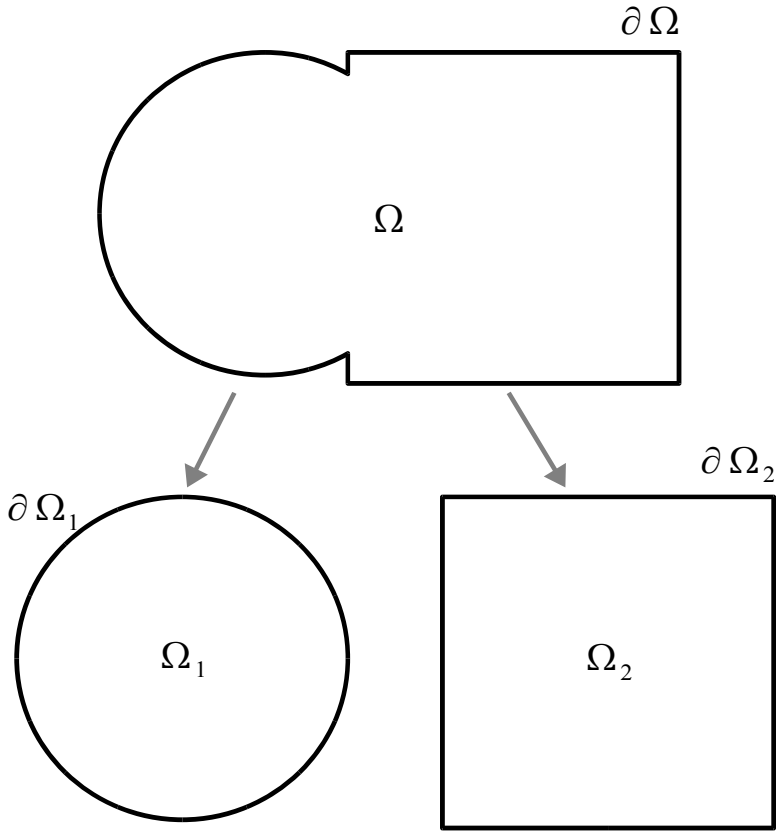
La soluciones obtenidas se intercambiarán con las particiones adyacentes y así, de forma iterativa, se irá aproximando la solución global del sistema.



Ejemplo tridimensional de descomposición de dominio

Método alternante de Schwarz

El algoritmo es conocido como el método alternante de Schwarz en paralelo.



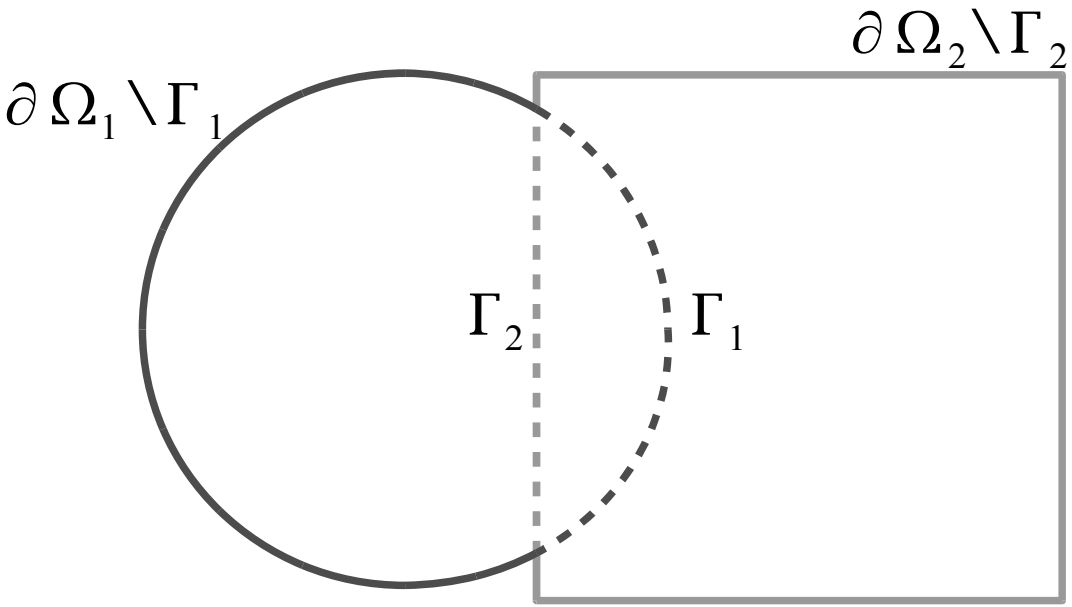
Particionamiento del dominio

Tenemos un dominio Ω con frontera $\partial\Omega$.

Sea $\boldsymbol{L}$ un operador diferencial tal que $\boldsymbol{L}\,\boldsymbol{x} = \boldsymbol{y}$ en Ω.

Se tienen condiciones Dirichlet $\boldsymbol{x} = \boldsymbol{b}$ en $\partial\Omega$

El dominio va a ser dividido en dos particiones Ω_1 y Ω_2 con fronteras $\partial\Omega_1$ y $\partial\Omega_2$ respectivamente.



Particiones traslapadas

Las particiones se traslapan, así $\Omega = \Omega_1 \cup \Omega_2$.

Las fronteras Γ_1 y Γ_2 son fronteras artificiales y son la parte de las fronteras de Ω_1 y Ω_2 que están en el interior de Ω.

El método alternante de Schwarz consiste en resolver de cada partición de forma independiente, fijando condiciones de Dirichlet en las fronteras artificiales de cada partición con los valores de la iteración previa de la partición adyacente.

$\boldsymbol{x}_1^0$, $\boldsymbol{x}_2^0$ aproximaciónes iniciales

ε tolerancia

$i \leftarrow 0$ número de iteración

mientras $\left\|\boldsymbol{x}_1^i - \boldsymbol{x}_1^{i-1}\right\| > \varepsilon$ o $\left\|\boldsymbol{x}_2^i - \boldsymbol{x}_2^{i-1}\right\| > \varepsilon$

resolver		resolver	
$\boldsymbol{L}\,\boldsymbol{x}_1^i = \boldsymbol{y}$	en Ω_1	$\boldsymbol{L}\,\boldsymbol{x}_2^i = \boldsymbol{y}$	en Ω_2
con $\boldsymbol{x}_1^i = \boldsymbol{b}$	en $\partial\boldsymbol{\Omega}_1 \setminus \boldsymbol{\Gamma}_1$	con $\boldsymbol{x}_2^i = \boldsymbol{b}$	en $\partial\boldsymbol{\Omega}_2 \setminus \boldsymbol{\Gamma}_2$
$\boldsymbol{x}_1^i \leftarrow \boldsymbol{x}_2^{i-1}\big\vert_{\Gamma_1}$	en $\boldsymbol{\Gamma}_1$	$\boldsymbol{x}_2^i \leftarrow \boldsymbol{x}_1^{i-1}\big\vert_{\Gamma_2}$	en $\boldsymbol{\Gamma}_2$

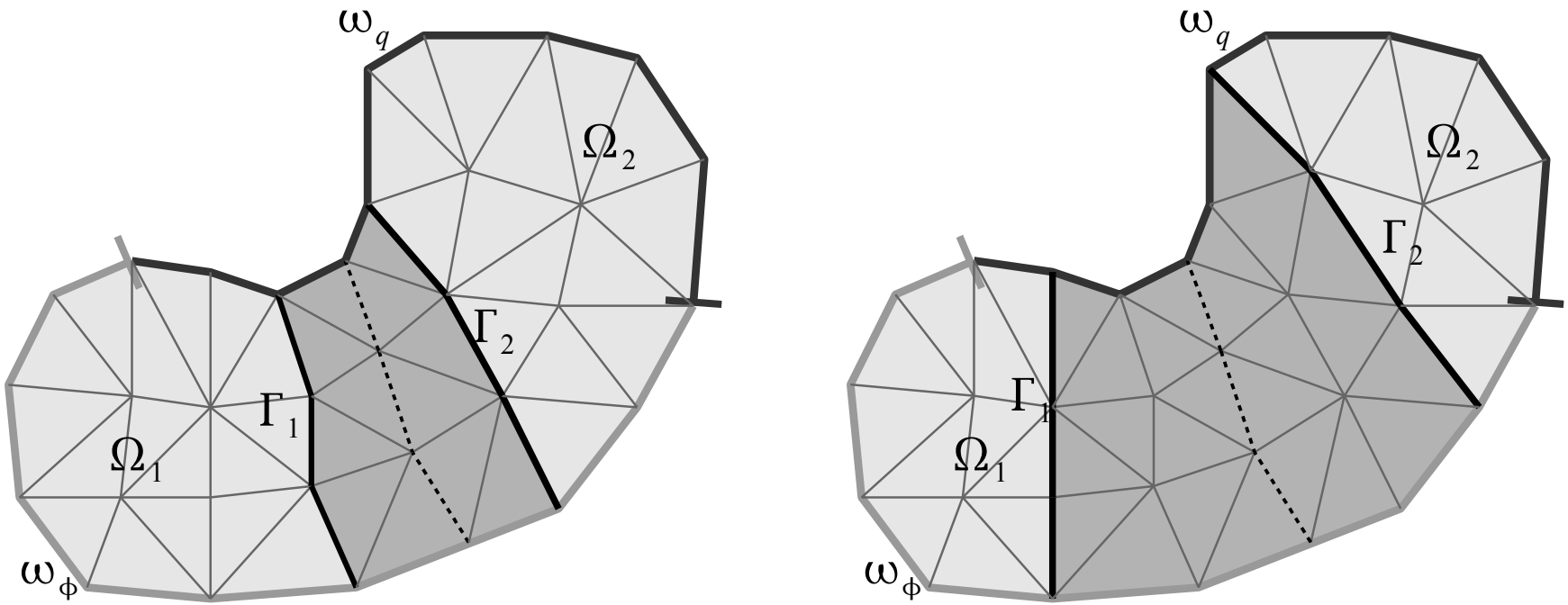
$i \leftarrow i+1$

Método alternante de Schwarz en paralelo

Cuando el operador $\boldsymbol{L}$ tiene una representación como matriz, el algoritmo de Schwarz corresponde a una generalización (debido al traslape) del método iterativo tradicional Gauss-Seidel por bloques.

Implementación con elemento finito

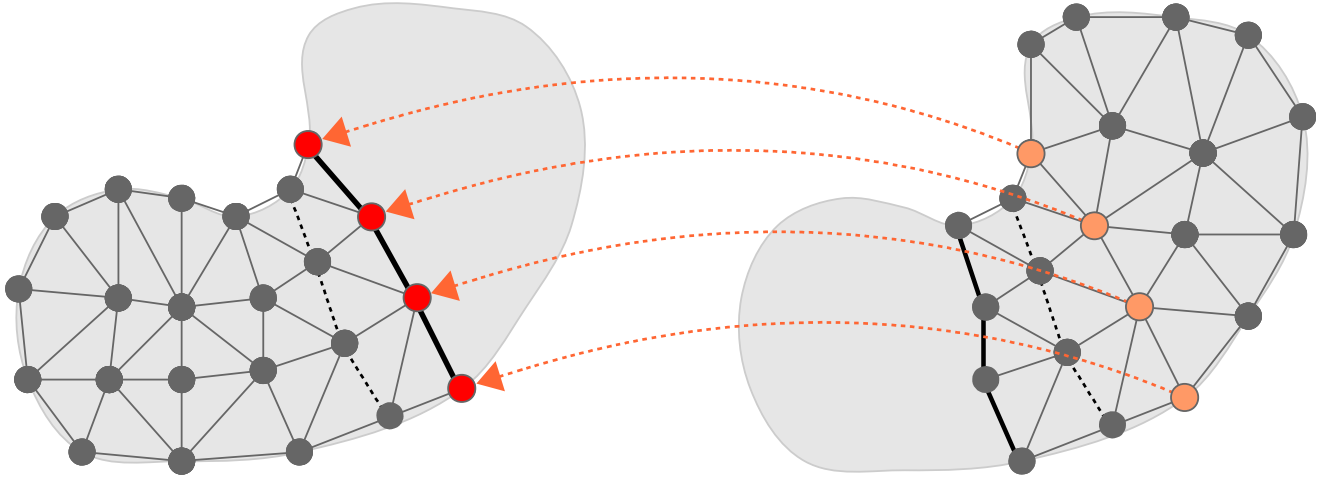
En problemas de elemento finito, el traslape se realiza aumentando a cada partición elementos de la partición adyacente a partir de la frontera entre las particiones.



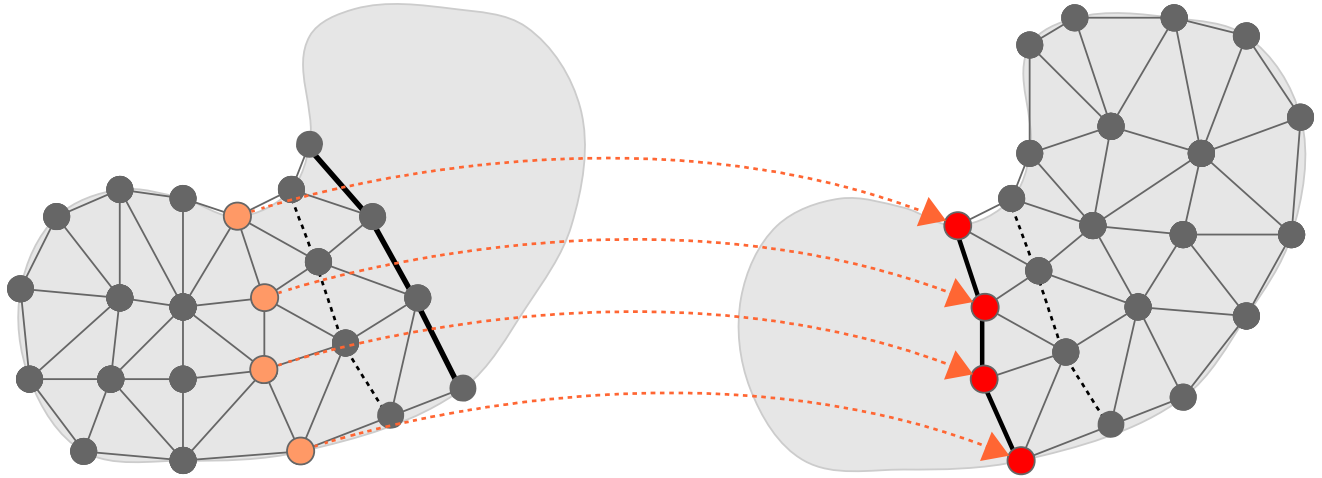
Dos tipos de traslape para particiones en un problema de elemento finito, a la izquieda con 1 nivel de traslape, a la derecha con 2 niveles de traslape.

Un tratamiento profundo de la teoría de los algoritmos de Schwarz puede consultarse en [Tose05].

Los *nodos fantasma* ● nos sirven para actualizar a los *nodos en la frontera artificial* ●.



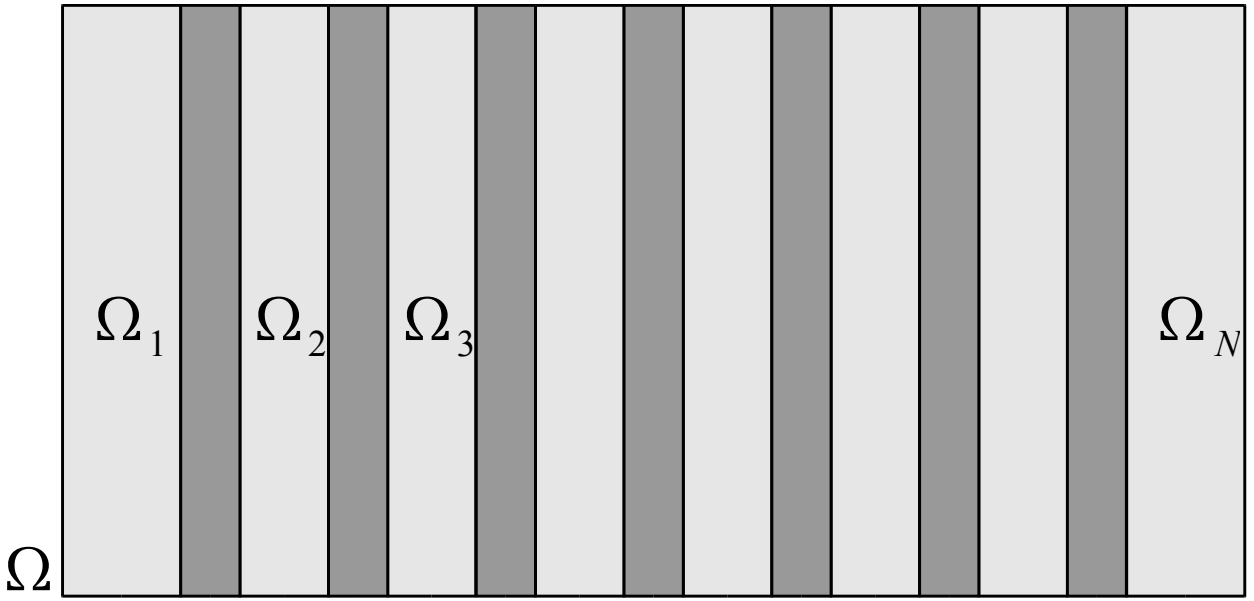
Entonces, con los resultados de la partición de la derecha vamos a definir condiciones de frontera de Dirichlet en la partición de la izquierda.



De forma simultánea se hace lo correspondiente para fijar las condiciones en la frontera artificial de la partición de la derecha.

Velocidad de convergencia

La velocidad de convergencia al utilizar descomposición de dominios se deteriora conforme se aumenta el número de particiones.



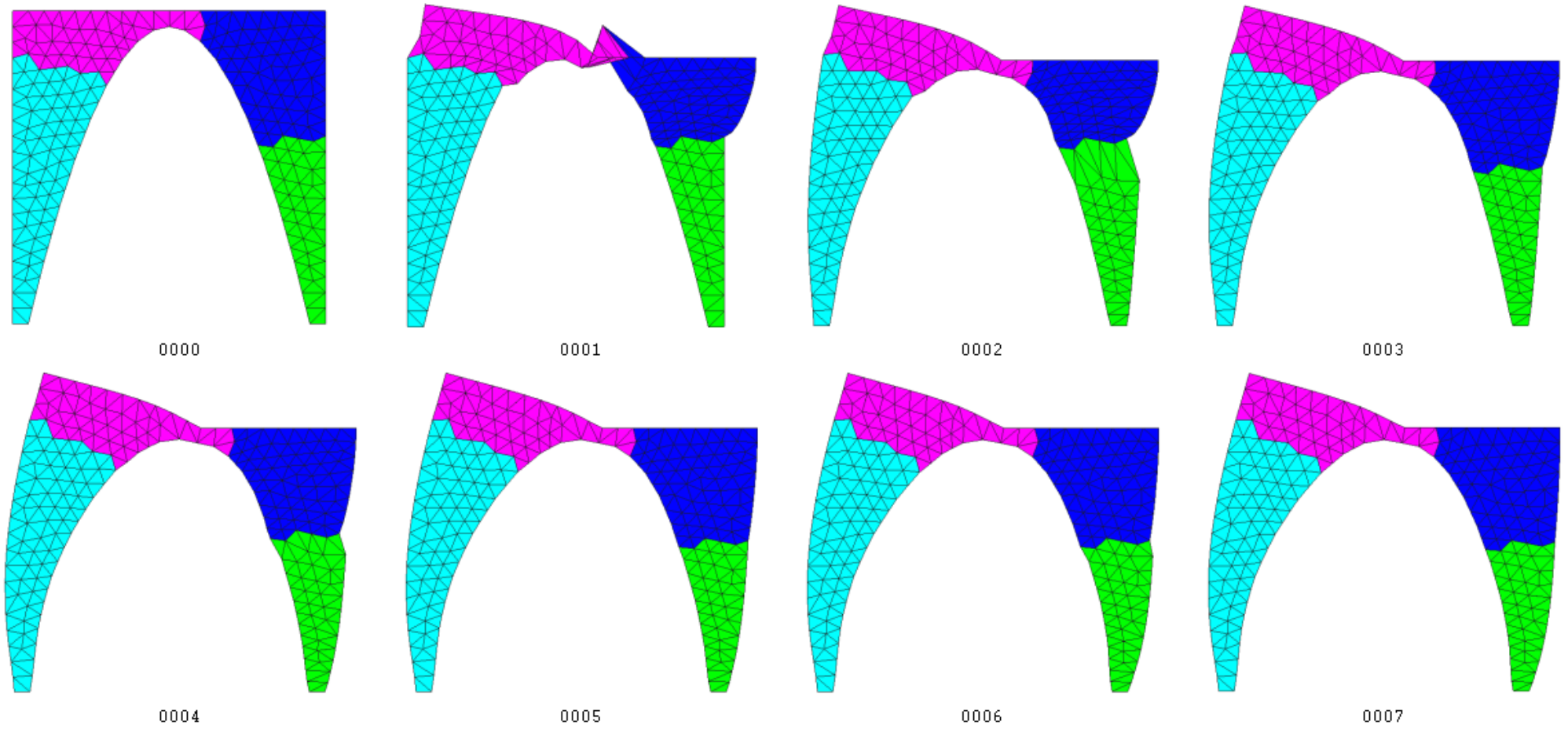
Caso patológico de descomposición de un dominio

En cada iteración del metodo alternante de Schwarz solo transferirá información entre las particiones adyacentes. Entonces, si se tiene una condición de frontera diferente de cero en la primer partición, y se inicia en la iteración 0, le tomará N iteraciones para que la solución local en la partición N sea diferente de cero.

Cuando el traslape aumenta se reduce el número de iteraciones necesarias para la convergencia.

Ejemplo de deformación de sólidos

Las siguientes imágenes muestran la evolución de un problema de deformación de sólidos en dos dimensiones utilizando el método alternante de Schwarz.

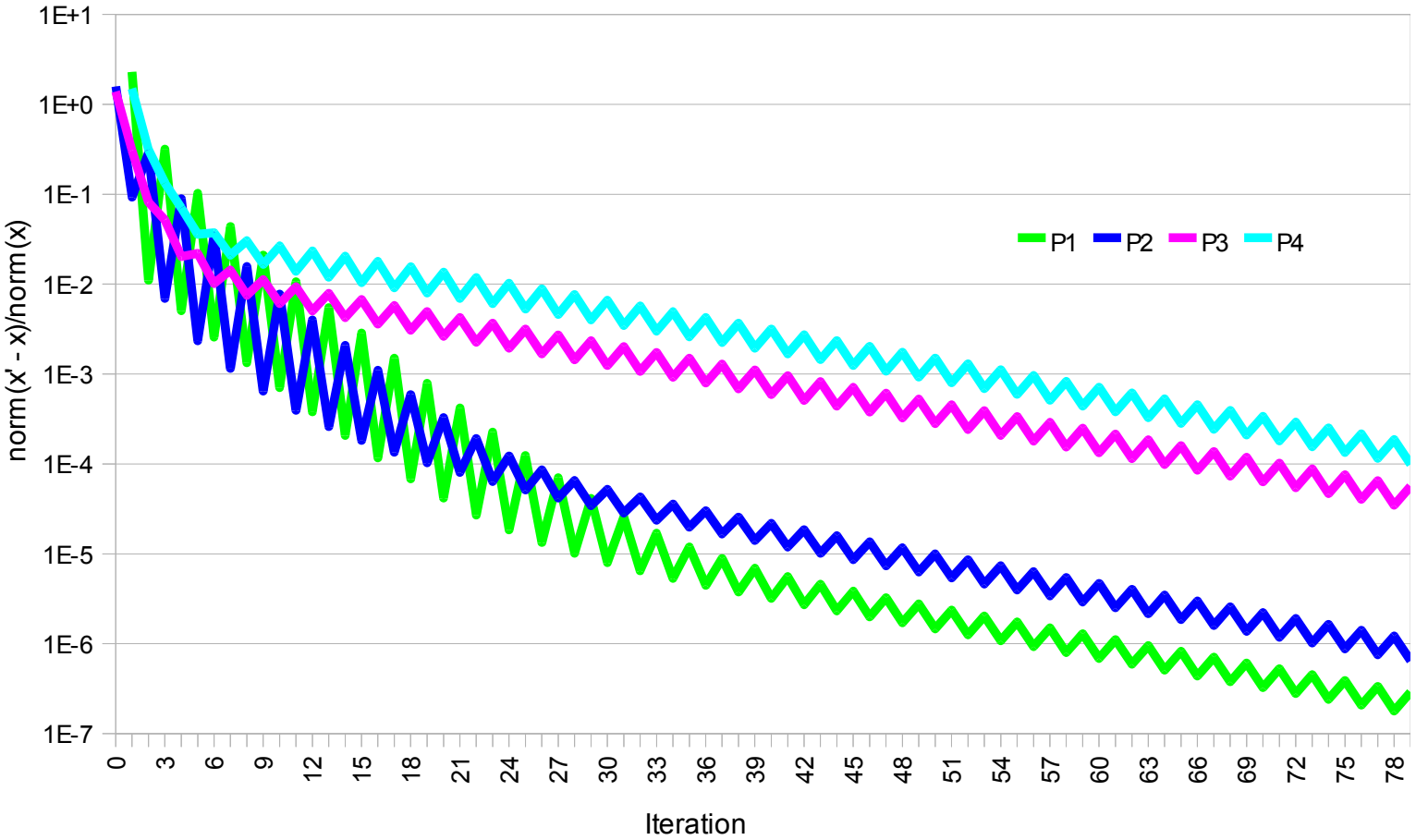


Primeras siete iteraciones del método alternante de Schwarz

La gráfica siguiente muestra la convergencia d^i de cada partición, medida como la norma ponderada de la diferencia entre la solución actual y la anterior

$$d^i = \frac{\left\| \boldsymbol{u}_{t-1}^i - \boldsymbol{u}_t^i \right\|}{\left\| \boldsymbol{u}_t^i \right\|},$$

donde $\boldsymbol{u}_t^i$ es el vector de desplazamiento resultante de resolver el sistema de ecuaciones de la partición i.



Usando la librería METIS para particionar mallas

METIS contiene varias rutinas para particionar grafos.
http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

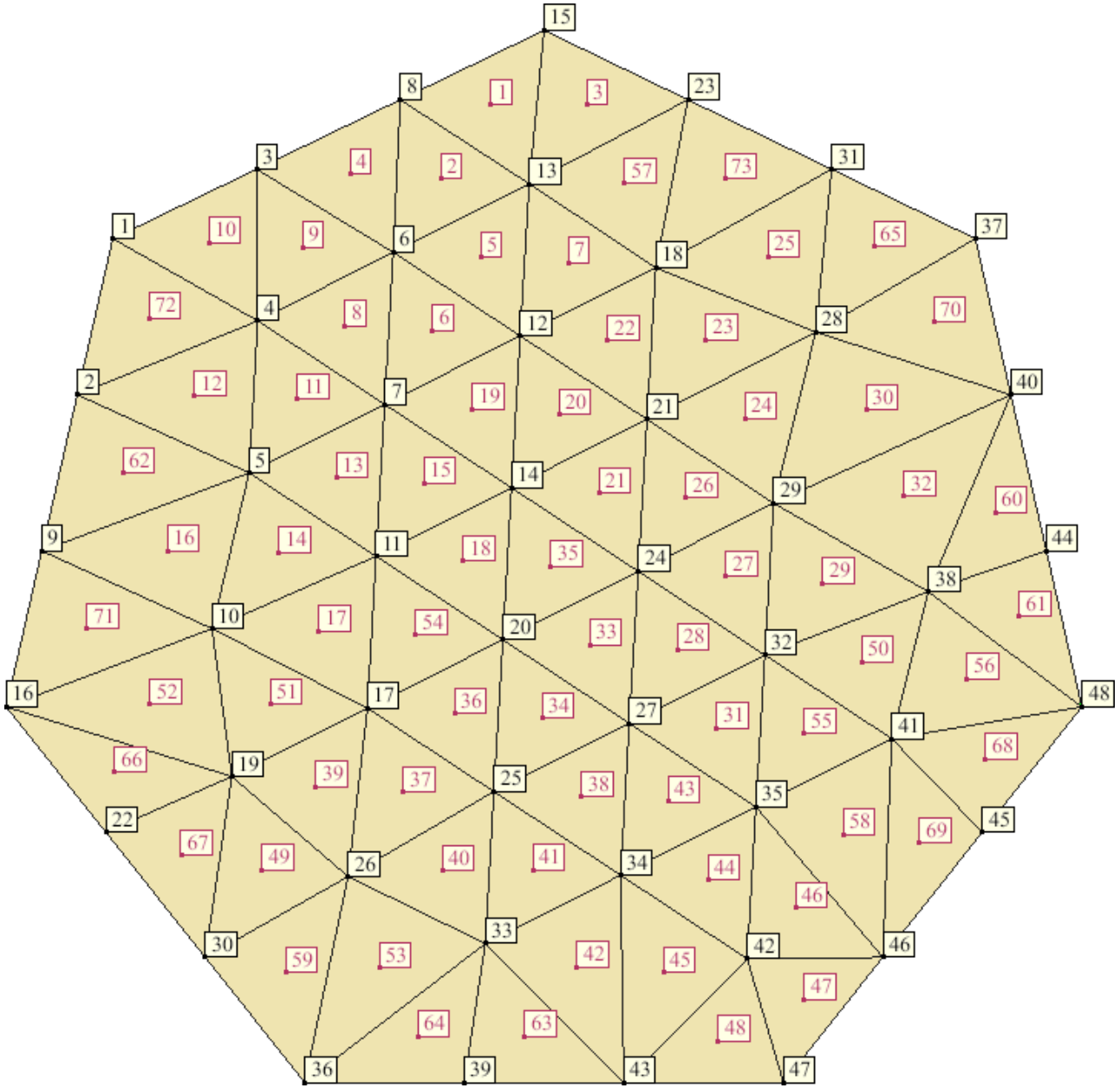
Vamos a utlizar la función **METIS_PartMeshNodal**.

void **METIS_PartMeshNodal**(int* ***ne***, int* ***nn***, idxtype* ***elmnts***, int* ***etype***, int* ***numflag***, int* ***nparts***, int* ***edgecut***, idxtype* ***epart***, idxtype* ***npart***)

Esta función divide una malla en *nparts*, utilizando como descripción de la malla la matriz de conectividades. Utiliza el algoritmo multinivel de disección anidada.

ne	Número de elementos en la malla.
nn	Número de nodos en la malla.
elmnts	Matriz de conectividades.
etype	Tipo de elemento: 1 triángulos, 2 tetrahedros, 3 hexaedros, 4 cuadriláteros.
numflag	Inidica si la numeración de los nodos (o elementos) comienza en 0 o en 1.
nparts	Número de particiones en las cuales se dividirá la malla.
edgecut	Si la malla se particionó con éxito, este valor indicará la cantidad de vertices que se dividieron.
epart	Vector de tamaño ne, indica a que partición pertenece cada elemento.
npart	Vector de tamaño nn, indica a que partición pertenece cada nodo. En nuestro caso no es muy útil, ya que un nodo puede pertenecer a más de una partición.

Ejemplo, dividir una geometría en 3 particiones



El siguiente código contiene los datos de conectividades para llamar al METIS.

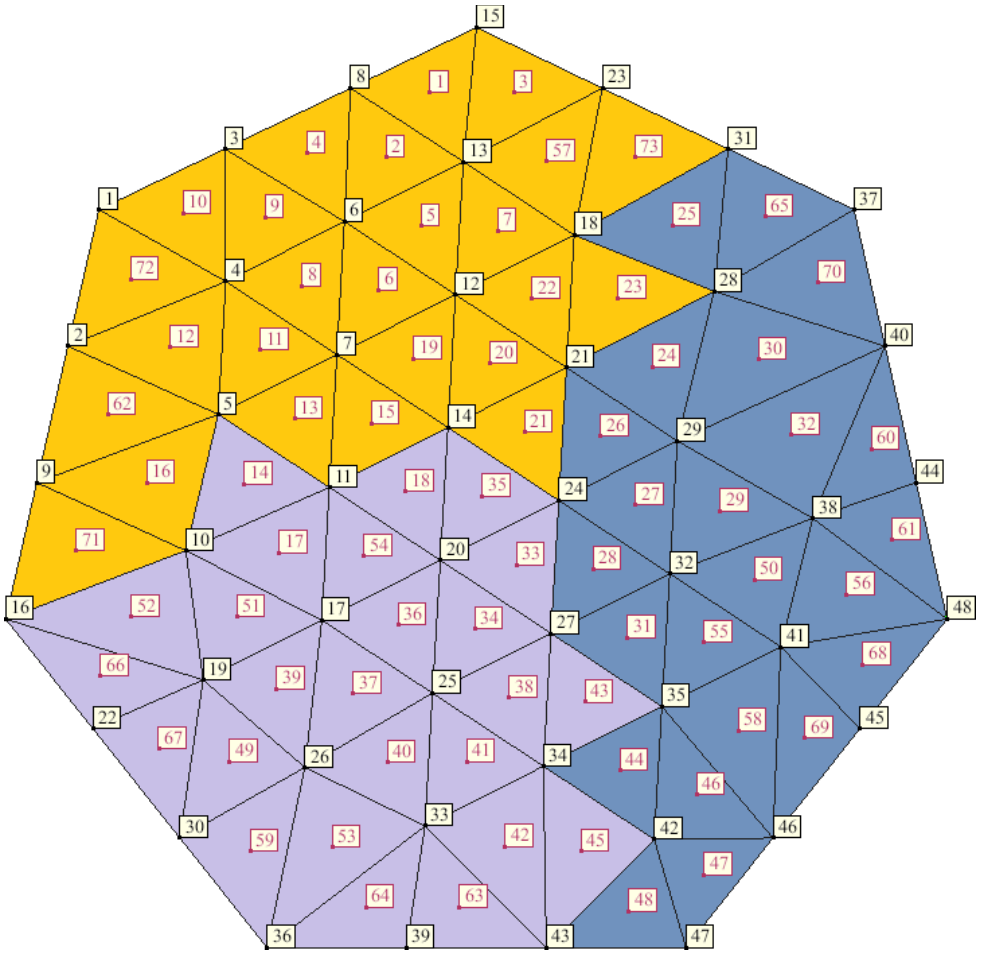
```
extern "C"{
     #include <metis.h>
}

int main()
{
     int ne = 73; // Number of elements
     int nn = 48; // Number of nodes
     idxtype elmnts[73][3] = // Connectivity matrix 73 rows by 3 columns
     {
               15,  8, 13,         13,  8,  6,         15, 13, 23,          6,  8,  3,          13,  6, 12,
               12,  6,  7,         13, 12, 18,          7,  6,  4,          4,  6,  3,           4,  3,  1,
                7,  4,  5,          5,  4,  2,          7,  5, 11,         11,  5, 10,           7, 11, 14,
               10,  5,  9,         11, 10, 17,         14, 11, 20,          7, 14, 12,          12, 14, 21,
               21, 14, 24,         12, 21, 18,         18, 21, 28,         28, 21, 29,          18, 28, 31,
               29, 21, 24,         29, 24, 32,         32, 24, 27,         29, 32, 38,          28, 29, 40,
               32, 27, 35,         29, 38, 40,         27, 24, 20,         27, 20, 25,          20, 24, 14,
               25, 20, 17,         25, 17, 26,         27, 25, 34,         26, 17, 19,          25, 26, 33,
               34, 25, 33,         34, 33, 43,         27, 34, 35,         35, 34, 42,          42, 34, 43,
               35, 42, 46,         46, 42, 47,         47, 42, 43,         26, 19, 30,          38, 32, 41,
               19, 17, 10,         19, 10, 16,         33, 26, 36,         11, 17, 20,          32, 35, 41,
               38, 41, 48,         13, 18, 23,         41, 35, 46,         30, 36, 26,          44, 40, 38,
               44, 38, 48,          2,  9,  5,         39, 43, 33,         39, 33, 36,          37, 31, 28,
               16, 22, 19,         19, 22, 30,         45, 48, 41,         45, 41, 46,          40, 37, 28,
                9, 16, 10,          1,  2,  4,         31, 23, 18
     };
     int etype = 1;      // Triangle
     int numflag = 1;    // Numeration starts with 1
     int nparts = 3;     // 3 partitions
     int edgecut;        // It will store edges divided
     idxtype epart[73]; // Element partition
     idxtype npart[48]; // Node partition
     METIS_PartMeshNodal(&ne, &nn, (idxtype*)elmnts, &etype, &numflag, &nparts, &edgecut, epart, npart);
     return 0;
}
```

El resultado es:

```
epart = {3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 2, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, 3, 2, 2, 1, 2, 2, 1, 1,
1, 3, 3, 3}
npart = {3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 2, 2, 3, 2, 3, 1, 2, 2, 2, 1, 1, 2, 1, 1, 2,
2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1}
```

O lo que es lo mismo, los elementos de cada partición serán:



Partición 1:

24, 25, 26, 27, 28, 29, 30, 31, 32, 44, 46, 47, 48, 50, 55, 56, 58, 60, 61, 65, 68, 69, 70

Partición 2:

14, 17, 18, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 49, 51, 52, 53, 54, 59, 63, 64, 66, 67

Partición 3:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 19, 20, 21, 22, 23, 57, 62, 71, 72, 73

Generación del traslape

El primer paso es que a partir de la matriz de conectividades ***elmnts*** y utilizando ***epart***, se pueden construir las matrices de conectividades para cada partición:

e	n_1	n_2	n_3
1	15	8	13
2	13	8	6
3	15	13	23
4	6	8	3
5	13	6	12
6	12	6	7
7	13	12	18
8	7	6	4
9	4	6	3
10	4	3	1
11	7	4	5
12	5	4	2
13	7	5	11
14	11	5	10
15	7	11	14
16	10	5	9
17	11	10	17
18	14	11	20
19	7	14	12
20	12	14	21
21	21	14	24
22	12	21	18
23	18	21	28
24	28	21	29
25	18	28	31
26	29	21	24
27	29	24	32
28	32	24	27
29	29	32	38
30	28	29	40
31	32	27	35
32	29	38	40
33	27	24	20
34	27	20	25
35	20	24	14
36	25	20	17
37	25	17	26
38	27	25	34
39	26	17	19
40	25	26	33
41	34	25	33
42	34	33	43
43	27	34	35
44	35	34	42
45	42	34	43
46	35	42	46
47	46	42	47
48	47	42	43
49	26	19	30
50	38	32	41
51	19	17	10
52	19	10	16
53	33	26	36
54	11	17	20
55	32	35	41
56	38	41	48
57	13	18	23
58	41	35	46
59	30	36	26
60	44	40	38
61	44	38	48
62	2	9	5
63	39	43	33
64	39	33	36
65	37	31	28
66	16	22	19
67	19	22	30
68	45	48	41
69	45	41	46
70	40	37	28
71	9	16	10
72	1	2	4
73	31	23	18

→

Partición 1

e	n_1	n_2	n_3
24	28	21	29
25	18	28	31
26	29	21	24
27	29	24	32
28	32	24	27
29	29	32	38
30	28	29	40
31	32	27	35
32	29	38	40
44	35	34	42
46	35	42	46
47	46	42	47
48	47	42	43
50	38	32	41
55	32	35	41
56	38	41	48
58	41	35	46
60	44	40	38
61	44	38	48
65	37	31	28
68	45	48	41
69	45	41	46
70	40	37	28

,

Partición 2

e	n_1	n_2	n_3
14	11	5	10
17	11	10	17
18	14	11	20
33	27	24	20
34	27	20	25
35	20	24	14
36	25	20	17
37	25	17	26
38	27	25	34
39	26	17	19
40	25	26	33
41	34	25	33
42	34	33	43
43	27	34	35
45	42	34	43
49	26	19	30
51	19	17	10
52	19	10	16
53	33	26	36
54	11	17	20
59	30	36	26
63	39	43	33
64	39	33	36
66	16	22	19
67	19	22	30

,

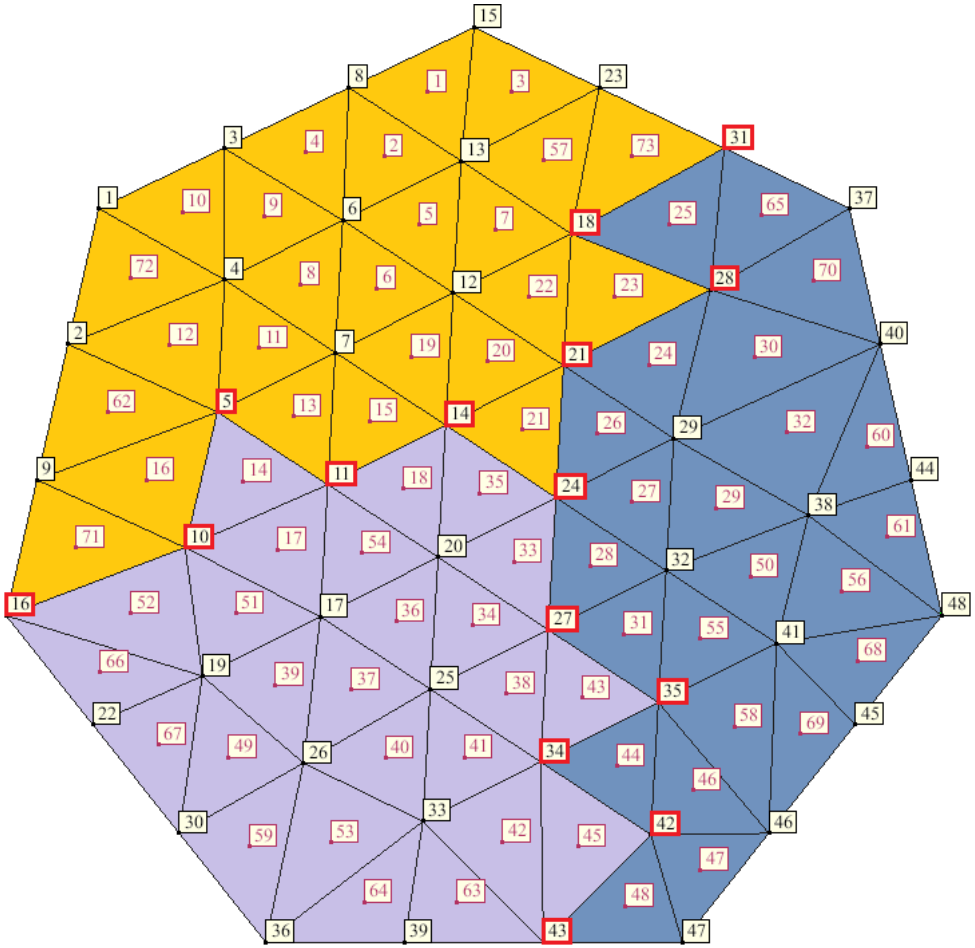
Partición 3

e	n_1	n_2	n_3
1	15	8	13
2	13	8	6
3	15	13	23
4	6	8	3
5	13	6	12
6	12	6	7
7	13	12	18
8	7	6	4
9	4	6	3
10	4	3	1
11	7	4	5
12	5	4	2
13	7	5	11
15	7	11	14
16	10	5	9
19	7	14	12
20	12	14	21
21	21	14	24
22	12	21	18
23	18	21	28
57	13	18	23
62	2	9	5
71	9	16	10
72	1	2	4
73	31	23	18

Detección de nodos en las fronteras entre las particiones

Hay que enlistar en qué matrices de conectividades aparece cada nodo.

n	Particiones		
1	3		
2	3		
3	3		
4	3		
5	2	3	
6	3		
7	3		
8	3		
9	3		
10	2	3	
11	2	3	
12	3		
13	3		
14	2	3	
15	3		
16	2	3	
17	2		
18	1	3	
19	2		
20	2		
21	1	3	
22	2		
23	3		
24	1	2	3
25	2		
26	2		
27	1	2	
28	1	3	
29	1		
30	2		
31	1	3	
32	1		
33	2		
34	1	2	
35	1	2	
36	2		
37	1		
38	1		
39	2		
40	1		
41	1		
42	1	2	
43	1	2	
44	1		
45	1		
46	1		
47	1		
48	1		



Los nodos marcados en rojo están en frontera.

Por ejemplo, el nodo 14 aparece en las conectividades de la particiones 2 y 3. Nodos que pertenezcan a más de una partición son *nodos frontera*.

Los nodos frontera de cada partición del ejemplo son:

Partición 1: 18, 21, 24, 27, 28, 31, 34, 35, 42, 43

Partición 2: 5, 10, 11, 14, 16, 24, 27, 34, 35, 42, 43

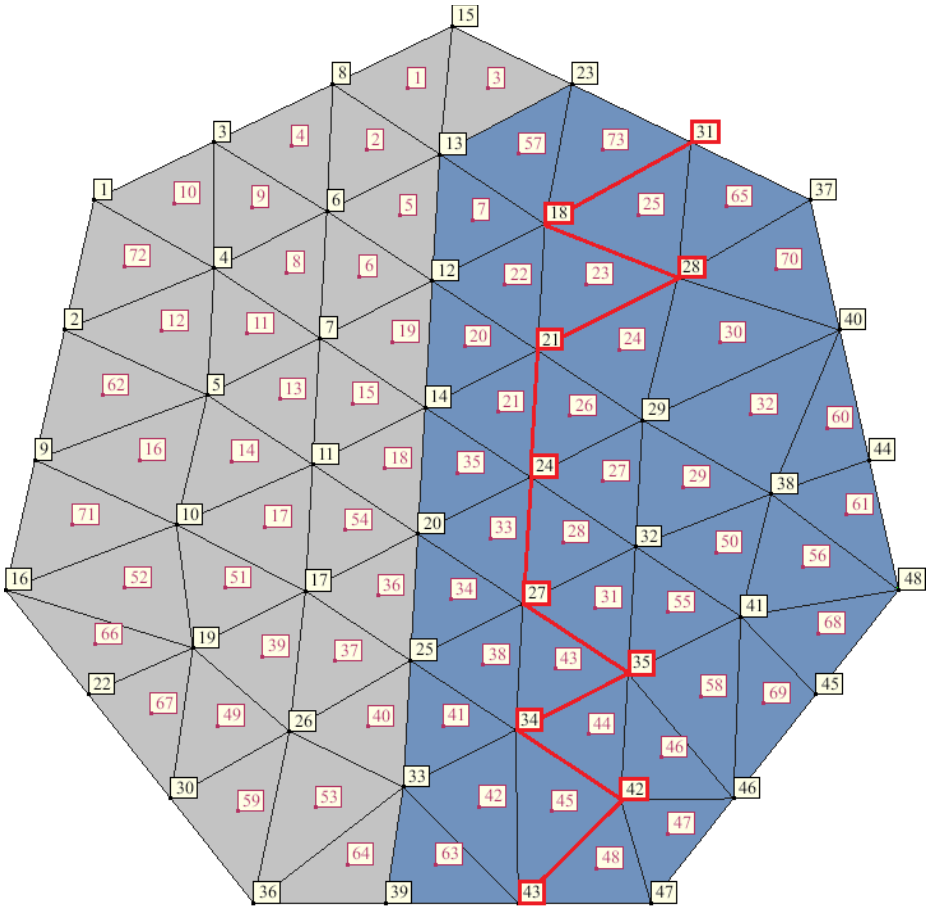
Partición 3: 5, 10, 11, 14, 16, 18, 21, 24, 28, 31

Extender un nivel la partición *i* significa agregar a la partición todos los elementos que en su conectividad tengan alguno de los nodos frontera de la partición i.

Por ejemplo, para la partición 1, los nodos frontera son: 18, 21, 24, 27, 28, 31, 34, 35, 42, 43.

Si observamos la matriz de conectividades original, el nodo 18 aparece en los elementos 7, 22, 23, 57 y 73. Todos estos elementos deben agregarse a la partición 1.

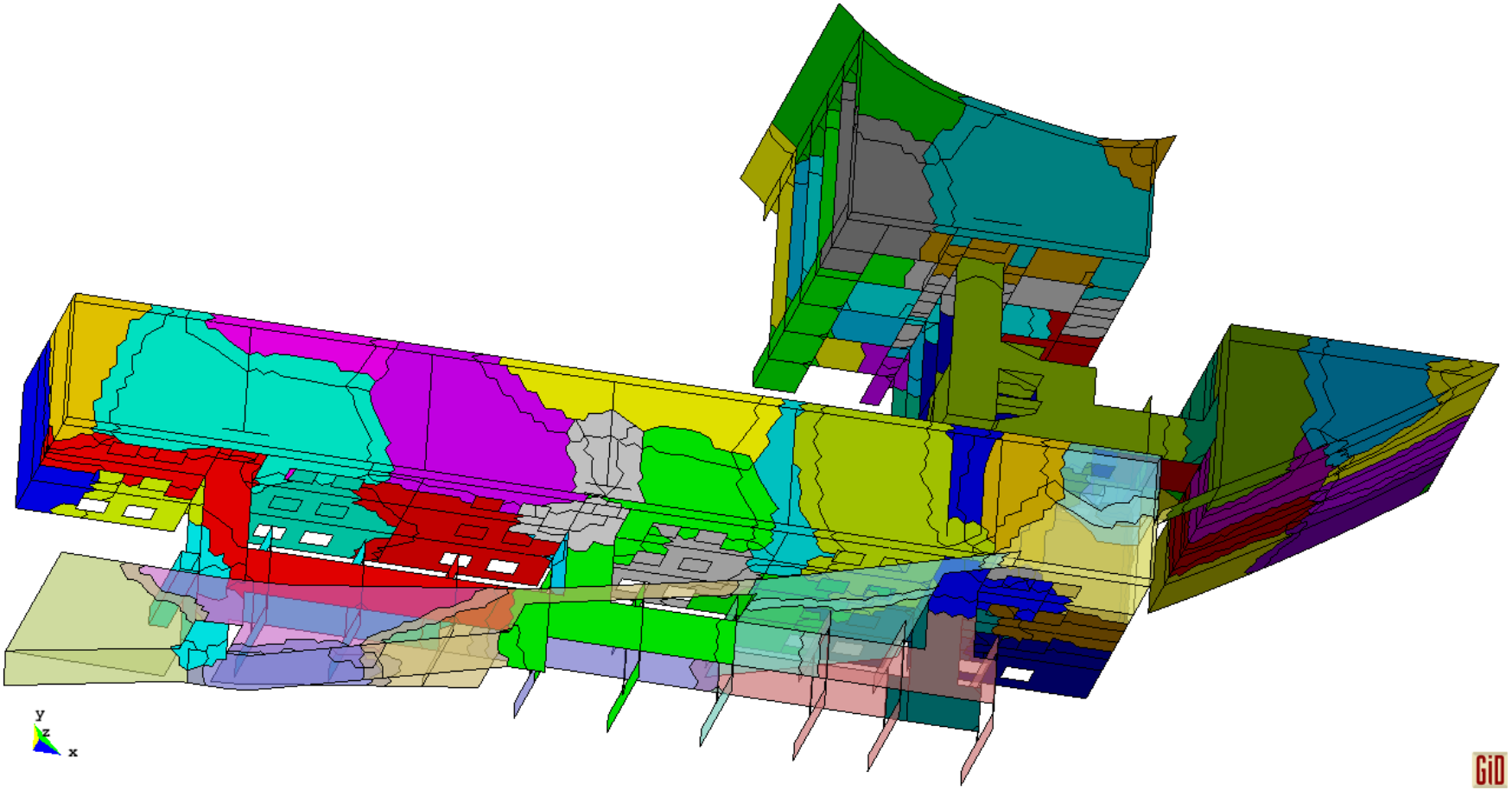
Este procedimiento se sigue para todos los nodos frontera 1. Así la partición queda extendida un nivel.



Este procedimiento se hace para cada partición.

Se puede repetir este procedimiento varias veces para agregar varias capas de traslape.

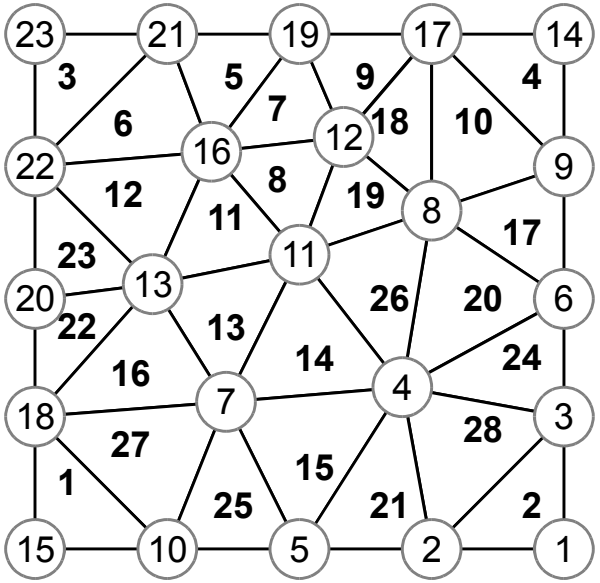
Aumentar las capas de traslape mejora la velocidad de convergencia, pero incrementa la memoria requerida para resolver cada sistema de ecuaciones.



Ejemplo de particionamiento con traslape de un problema en 3D.

Ejemplo con una geometría sencilla y dos particiones

Vamos a mostrar como se realiza la partición del dominio con traslape necesario para implementar el método alternante de Schwarz, utilizando para ello un ejemplo sencillo de un problema de elemento finito en dos dimensiones con una malla de 28 elementos triangulares y 23 nodos.

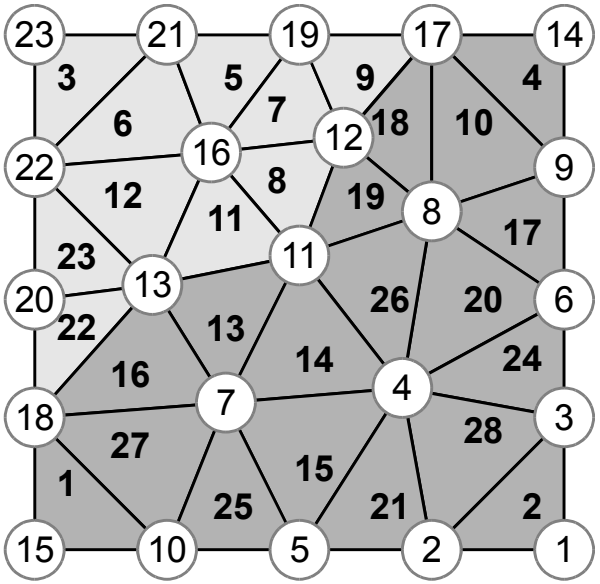


E	n_1	n_2	n_3
1	15	10	18
2	1	3	2
3	23	22	21
4	14	17	9
5	19	21	16
6	16	21	22
7	19	16	12
8	12	16	11
9	19	12	17
10	17	8	9
11	11	16	13
12	13	16	22
13	11	13	7
14	11	7	4

E	n_1	n_2	n_3
15	4	7	5
16	7	13	18
17	6	9	8
18	8	17	12
19	8	12	11
20	6	8	4
21	5	2	4
22	20	18	13
23	22	20	13
24	3	6	4
25	10	5	7
26	11	4	8
27	18	10	7
28	2	3	4

La matriz de conectividades es pasada a la función de partición de METIS. La figura siguiente muestra el resultado, éste indica a que partición pertenece cada elemento.

Indicaremos los elementos de la primer partición como EP_1 y los de la segunda como EP_2.

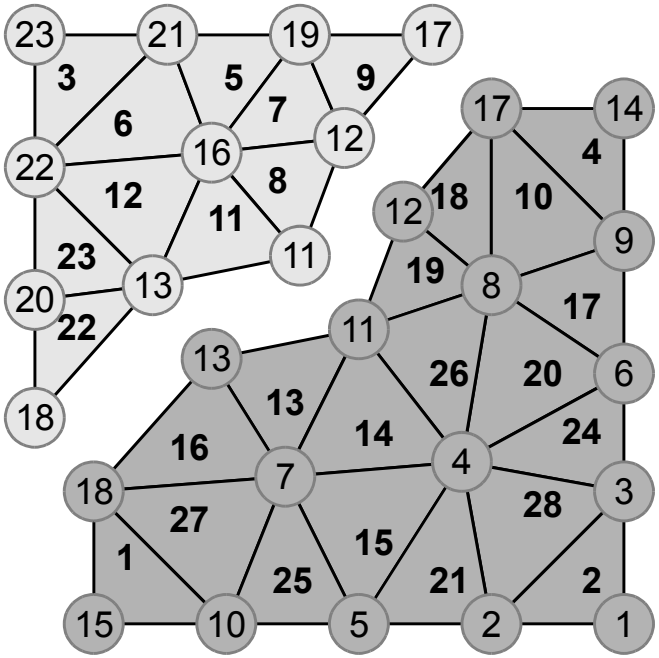


EP_1
3
5
6
7
8
9
11
12
22
23

EP_2	
1	18
2	19
4	20
10	21
13	24
14	25
15	26
16	27
17	28

Buscando en la tabla de conectividades determinamos que nodos pertenecen a cada elemento en cada partición.

Indicaremos los nodos de la primer partición como nP_1 y los de la segunda como nP_2.

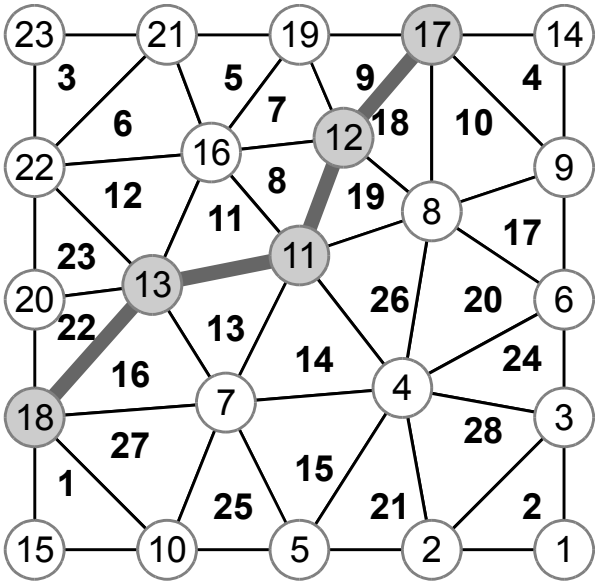


nP_1
11
12
13
16
17
18
19
20
21
22
23

nP_2	
1	10
2	11
3	12
4	13
5	14
6	15
7	17
8	18
9	

Para definir qué nodos pertencen a la frontera entre las particiones, buscamos los nodos que aparezcan en más de una partición.

En el caso de la figura siguiente que aparezcan tanto en nP_1 como en nP_2.

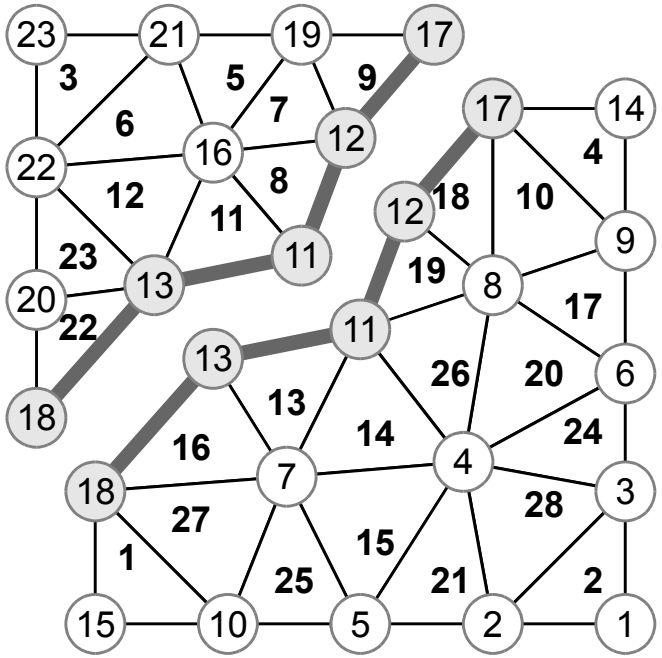


nP_1
11
12
13
16
17
18
19
20
21
22
23

nP_2	
1	10
2	11
3	12
4	13
5	14
6	15
7	17
8	18
9	

A partir de este momento dividiremos la malla en el número de particiones, la frontera encontrada anteriormente existirá en cada sub-malla.

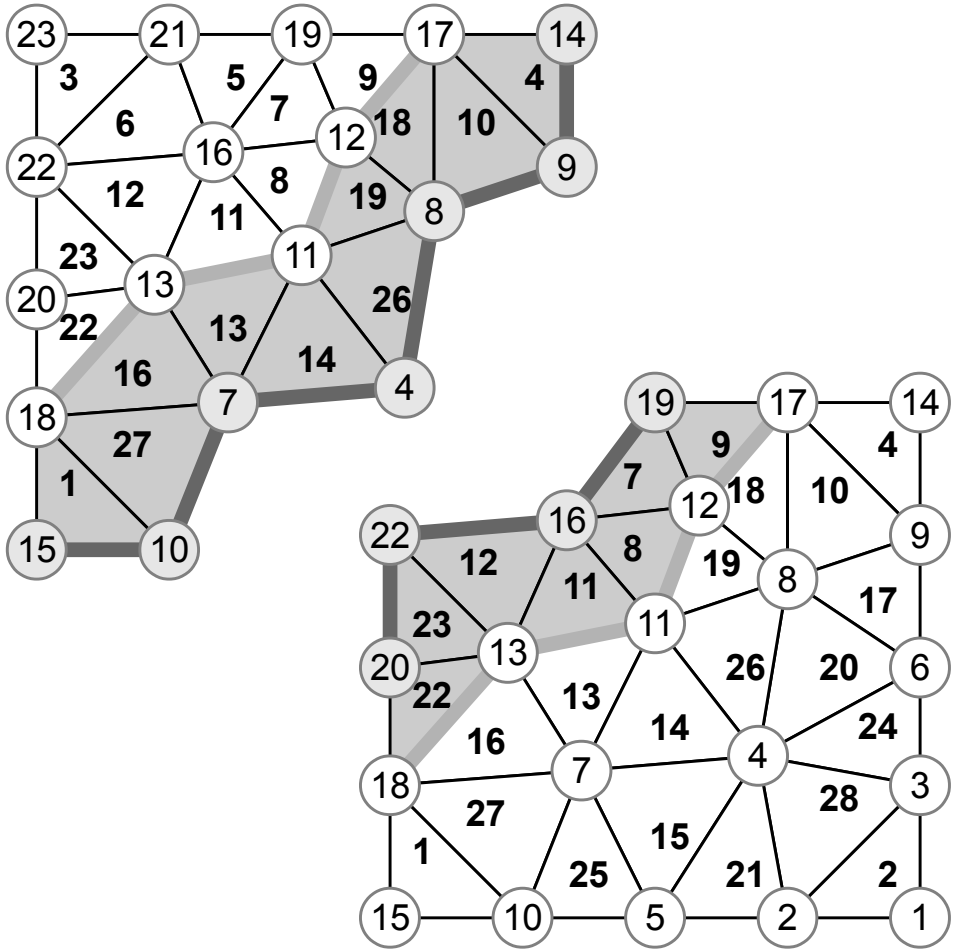
En nuestro ejemplo existen dos fronteras FP_1 y FP_2, en esta etapa ambas son iguales.



FP_1	FP_2
11	11
12	12
13	13
17	17
18	18

Aumentar una capa de traslape significa que se agregarán a la sub-partición los elementos que compartan nodo en la frontera, pero que no estén en dicha sub-partición.

Los nodos de estos nuevos elementos que no estén en la sub-partición se agregarán y formarán la nueva frontera, suplantando a la anterior.



FP_1		FP_2
4		16
7		19
8		20
9		22
10		
14		
15		

EP_1		EP_2
1	12	1 17
3	13	2 18
4	14	4 19
5	16	7 20
6	18	8 21
7	19	9 22
8	22	10 23
9	23	11 24
10	26	12 25
11	27	13 26
		14 27
		15 28
		16

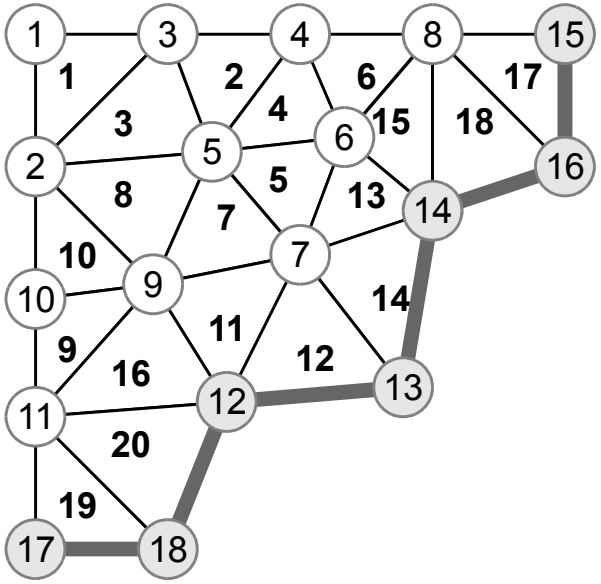
nP_1	
4	15
7	16
8	17
9	18
10	19
11	20
12	21
13	22
14	23

nP_2	
1	12
2	13
3	14
4	15
5	16
6	17
7	18
8	19
9	20
10	22
11	

En la figura se observa cómo se agrega una capa de traslape a cada partición, se denotan los elementos agregados y la nueva frontera formada.

Este proceso se puede repetir tantas veces como capas de traslape se deseen.

Ahora es necesaria la renumeración local a cada partición de elementos y nodos.



EP_1	E
1	3
2	5
3	6
4	7
5	8
6	9
7	11
8	12
9	22
10	23

EP_1	E
11	13
12	14
13	19
14	26
15	18
16	16
17	4
18	10
19	1
20	27

nP_1	n
1	23
2	22
3	21
4	19
5	16
6	12
7	11
8	17
9	13

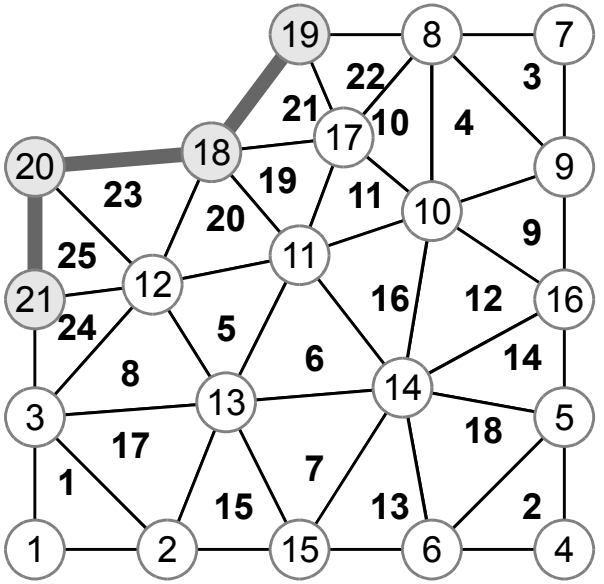
nP_1	n
10	20
11	18
12	7
13	4
14	8
15	14
16	9
17	15
18	10

EP_2	E
1	1
2	2
3	4
4	10
5	13
6	14
7	15
8	16
9	17
10	18
11	19
12	20
13	21

EP_2	E
14	24
15	25
16	26
17	27
18	28
19	8
20	11
21	7
22	9
23	12
24	22
25	23

nP_2	n
1	15
2	10
3	18
4	1
5	3
6	2
7	14
8	17
9	9
10	8
11	11

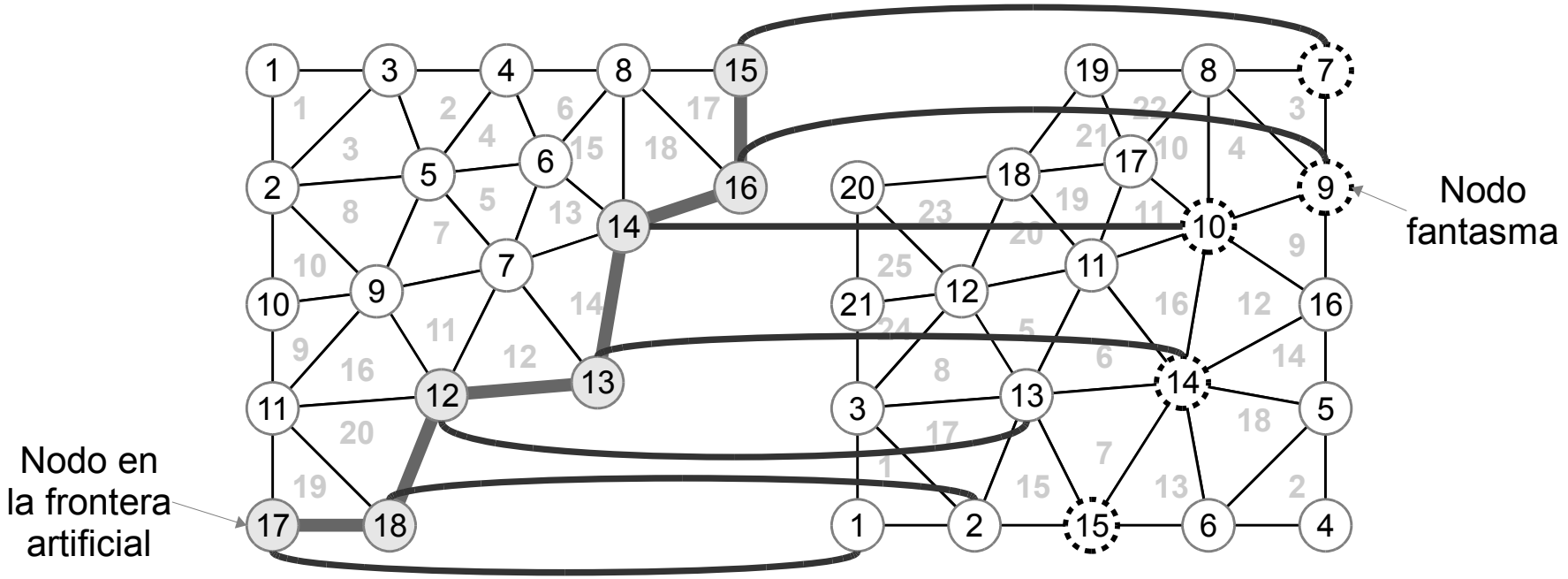
nP_2	n
12	13
13	7
14	4
15	5
16	6
17	12
18	16
19	19
20	22
21	20



La figura muestra el reordenamiento para el ejemplo, las tablas indican la numeración local y global, tanto de elementos como de nodos.

Cuando se utiliza el solver con factorización Cholesky será necesario agregar un paso extra que imponga una enumeración de los nodos más adecuada.

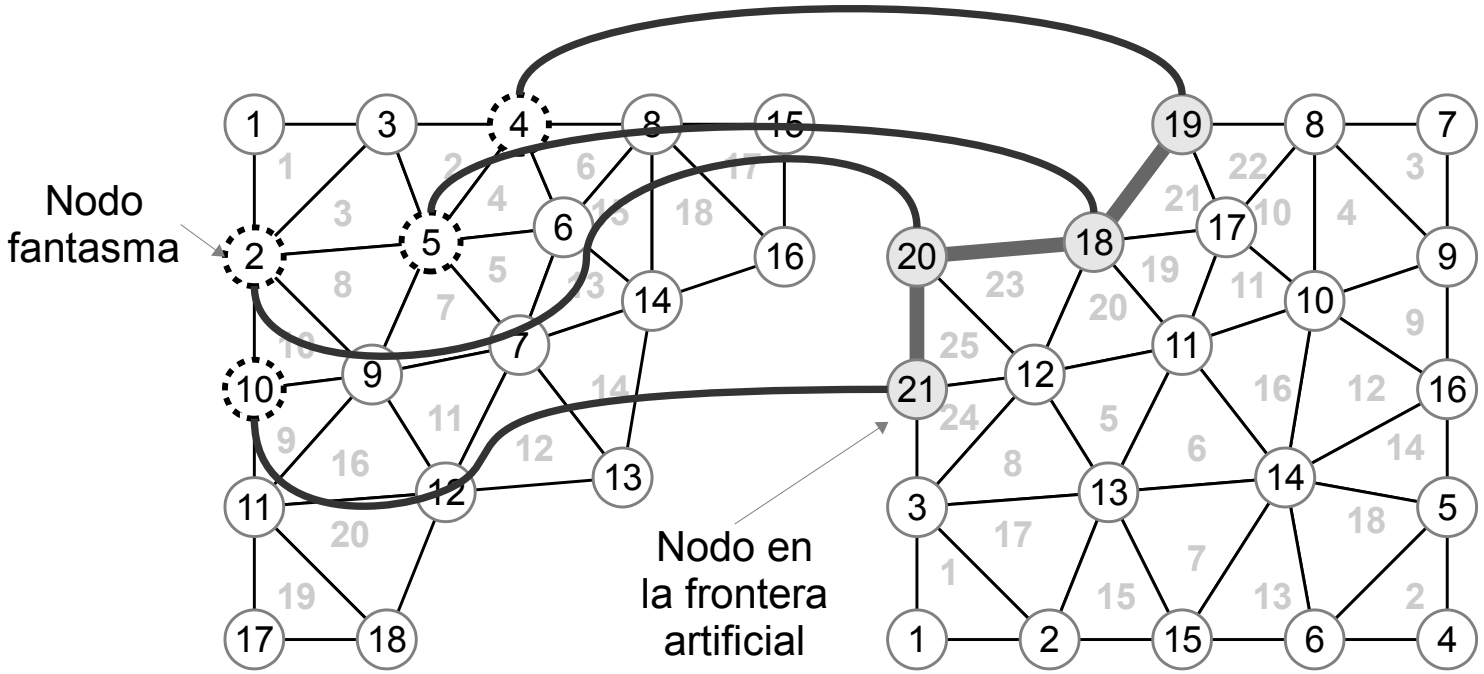
El paso final es crear enlaces entre los nodos en la frontera artificial de cada sub-malla con su correspondiente nodo fantasma en la otra sub-malla.



gP_1	nP_1	nP_2
1	15	7
2	16	9
3	14	10
4	13	14
5	12	13
6	18	2
7	17	1

En la figura se muestran los enlaces formados entre los nodos en la frontera artificial de la partición 1 y los nodos fantasma en la partición 2.

Cada enlace guarda dos datos, el índice del nodo en la frontera artificial con la numeración local y el índice del nodo fantasma con la numeración correspondiente en la partición adyacente.



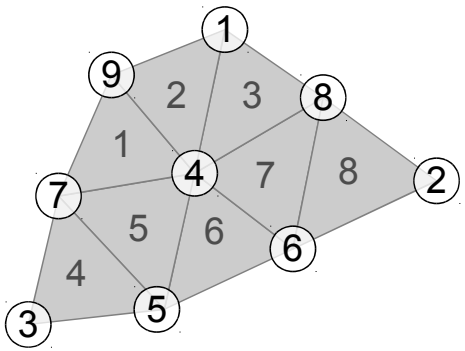
gP_2	nP_2	nP_1
1	19	4
2	18	5
3	20	2
4	21	10

En la figura se muestran los enlaces formados entre los nodos en la frontera artificial de la partición 2 y los nodos fantasma en la partición 1.

Para realizar particionamientos con más de dos particiones se requerirá guardar una lista de enlaces por cada partición con la que se tenga una frontera artificial.

Estructura de una matriz rala a partir de conectividades

Conocer de antemano las conectividades de los elementos hace posible generar la estructura la correspondiente matriz $\boldsymbol{A}$ en memoria.



E	N_1	N_2	N_3
1	9	4	7
2	1	4	9
3	1	8	4
4	7	5	3
5	4	5	7
6	4	6	5
7	8	6	4
8	8	2	6

1	8	4	9			
2	8	6				
3	7	5				
4	1	8	6	5	7	9
5	4	6	3	7		
6	8	2	4	5		
7	9	4	5			
8	1	2	4	6		
9	1	4	7			

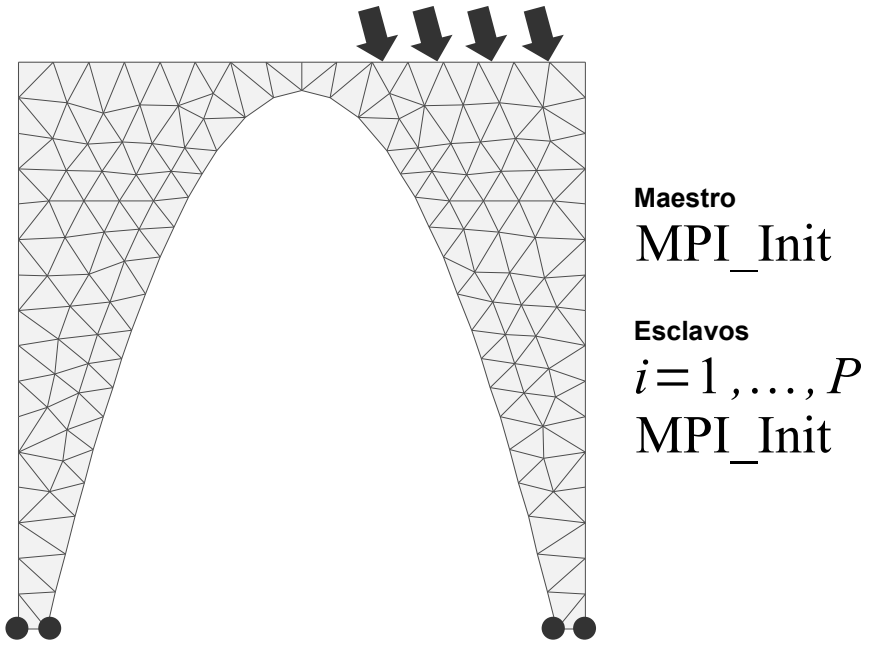
$$\begin{pmatrix} a_{11} & 0 & 0 & a_{14} & 0 & 0 & 0 & a_{18} & a_{19} \\ 0 & a_{22} & 0 & 0 & 0 & a_{26} & 0 & a_{28} & 0 \\ 0 & 0 & a_{33} & 0 & a_{35} & 0 & a_{37} & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} & a_{49} \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & 0 & 0 \\ 0 & a_{62} & 0 & a_{64} & a_{65} & a_{66} & 0 & a_{68} & 0 \\ 0 & 0 & a_{73} & a_{74} & a_{75} & 0 & a_{77} & 0 & a_{79} \\ a_{81} & a_{82} & 0 & a_{84} & 0 & a_{86} & 0 & a_{88} & 0 \\ a_{91} & 0 & 0 & a_{94} & 0 & 0 & a_{97} & 0 & a_{99} \end{pmatrix}$$

Malla, matriz de conectividades, tabla de adyacencias, matriz rala.

Ejemplo de implementación con MPI

Vamos a explicar de forma simplificada las rutinas de MPI [MPIF08] utilizadas en nuestra implementación del método alternante de Schwarz.

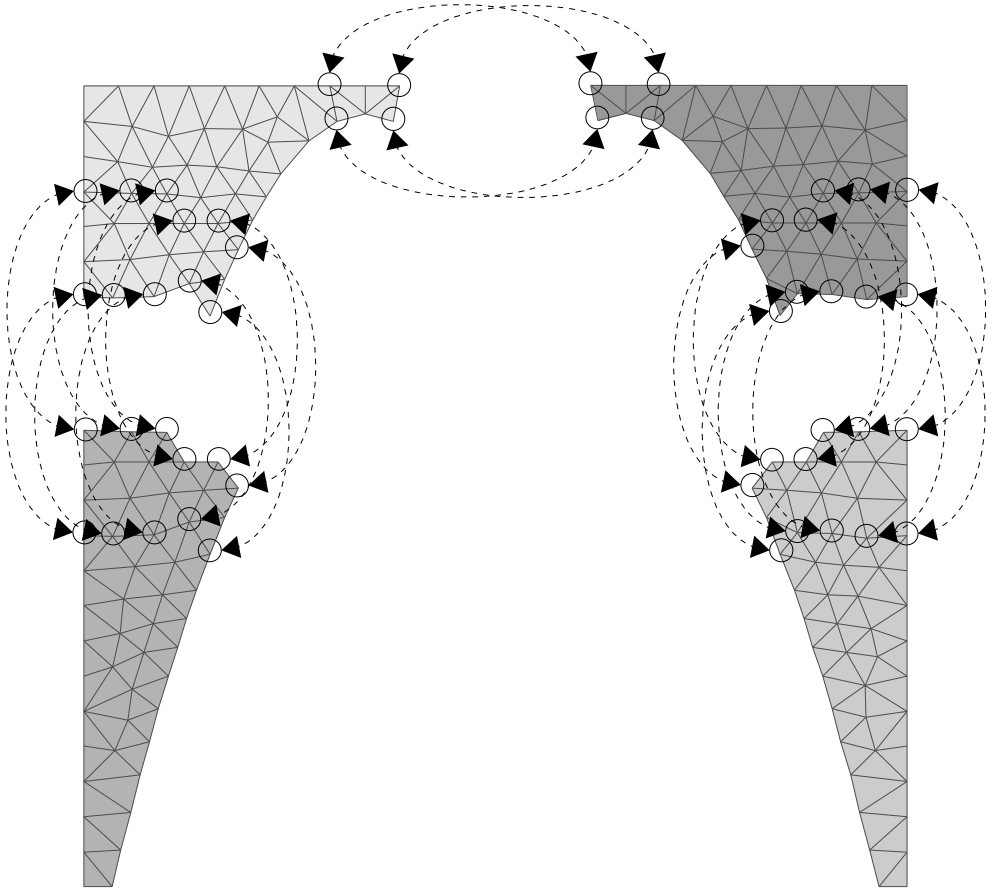
1. P que indica la cantidad de particiones. Para este ejemplo, el programa se instancia 5 veces, asignándose el proceso con rango 0 como el maestro y los procesos 1 a 4 como esclavos, es decir, tendremos 4 particiones. Cada nodo entra en el esquema de MPI llamando la rutina MPI_Init.



Entrada del programa.

2. En el nodo maestro se realiza un particionamiento con traslape.

Se generan las particiones con traslape, se reenumeran elementos y nodos, finalmente se generan tablas con los enlaces de los nodos fantasma.



Particionamiento del dominio con traslape.

3. El nodo maestro entra en un ciclo $i=1,\ldots,P$ en el cual se envian los siguientes datos correspondientes a cada partición i utilizando la función MPI_Send:

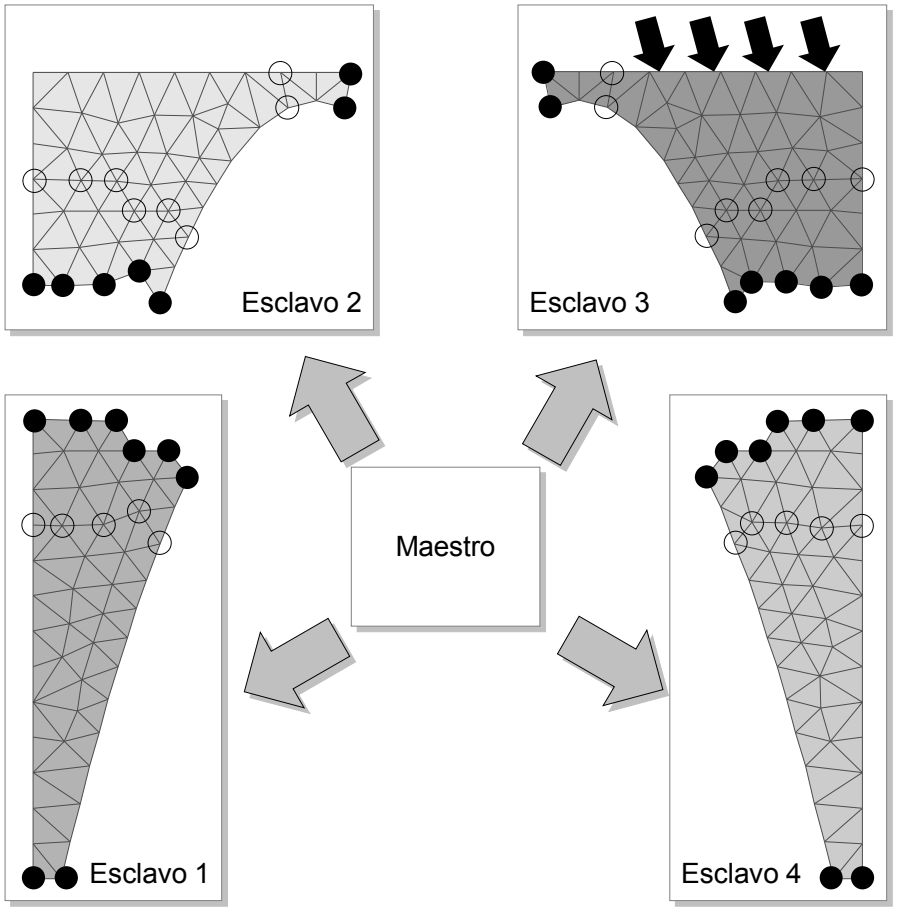
- $\boldsymbol{N}^i$ coordenadas de los nodos
- $\boldsymbol{E}^i$ tabla de conectividades de los elementos
- $\boldsymbol{C}^i$ condiciones de frontera

Se envian en un ciclo $j=1,\ldots,P$ con $j\neq i$:

- Los $\boldsymbol{L}^{ij}$ enlaces con los indices de los nodos de frontera artificial y su correspondiente nodo fantasma en la partición j.

Con la función MPI_Irecv$\left(i, s^i\right)$ se crea una petición sin bloqueo del estatus del nodo i, esto le permite al nodo maestro seguir enviando datos sin esperar a que los esclavos estén listos.

Cada esclavo generará su sistema de ecuaciones y cuando esté listo enviará su estatus s^i al nodo maestro.



Maestro

$$\text{MPI\_Send}\left(i, \boldsymbol{N}^i\right)$$
$$\text{MPI\_Send}\left(i, \boldsymbol{E}^i\right)$$
$$\text{MPI\_Send}\left(i, \boldsymbol{C}^i\right)$$
$$\text{MPI\_Send}\left(i, \boldsymbol{L}^{ij}\right)$$
$$\text{MPI\_Irecv}\left(i, s^i\right)$$
$$\text{MPI\_Waitall}\left(s^i\right)$$

Esclavos $i = 1, \ldots, P$

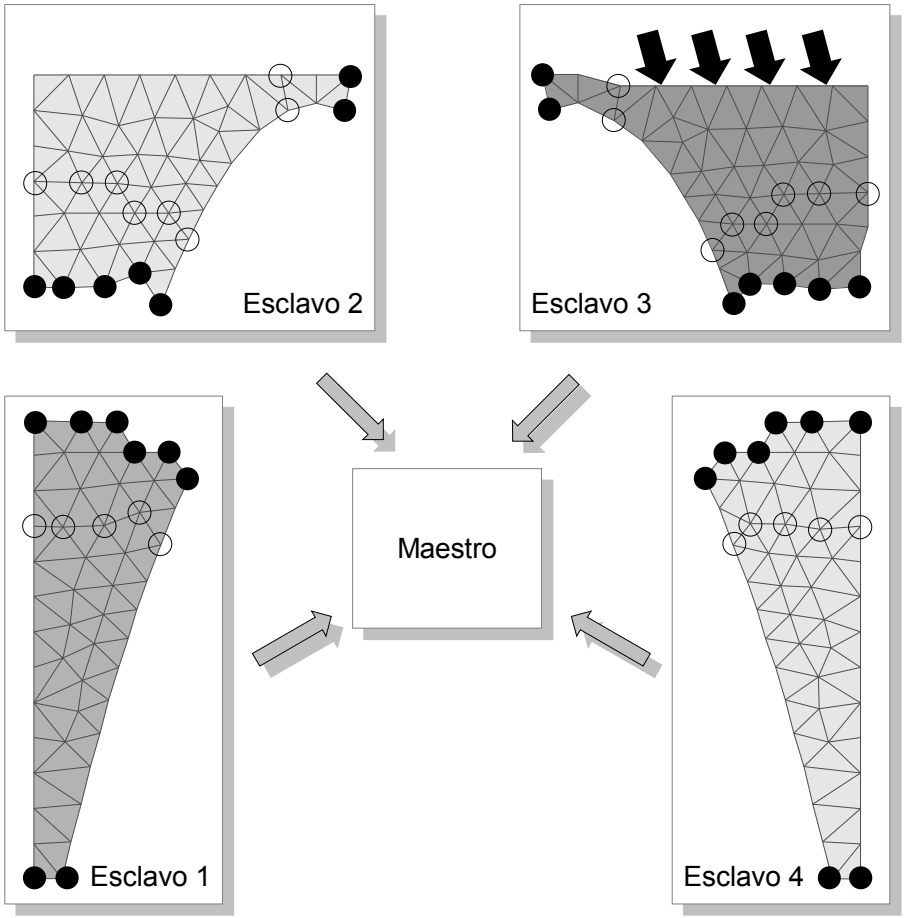
$$\text{MPI\_Recv}\left(0, \boldsymbol{N}^i\right)$$
$$\text{MPI\_Recv}\left(0, \boldsymbol{E}^i\right)$$
$$\text{MPI\_Recv}\left(0, \boldsymbol{C}^i\right)$$
$$\text{MPI\_Recv}\left(0, \boldsymbol{L}^{ij}\right)$$
$$\text{MPI\_Send}\left(0, s^i\right)$$

Creación de problemas individuales.

Con la función $\text{MPI\_Waitall}\left(s^i\right)$ el nodo maestro espera el estatus de todos los esclavos antes de continuar.

4. En este punto el programa comienza a iterar hasta que se logre la convergencia del algoritmo de Schwarz.

En cada esclavo el sistema de ecuaciones es resuelto localmente. Al terminar cada esclavo envía la diferencia entre la solución anterior y la actual d^i al nodo maestro, el cual evalua la convergencia global del problema.



Maestro

$\text{MPI\_Irecv}\left(i, d^i\right)$

$\text{MPI\_Waitall}\left(d^i\right)$

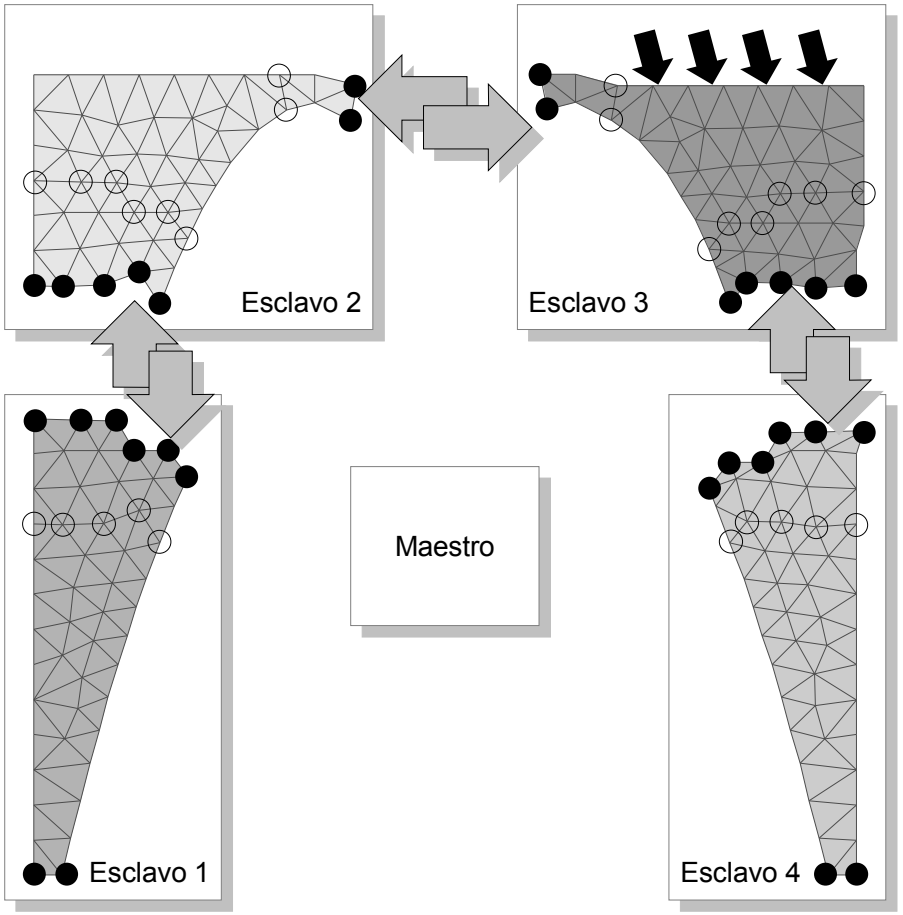
Esclavos $i = 1, \ldots, P$

$\text{MPI\_Send}\left(0, d^i\right)$

Reporte del estado de la convergencia al nodo maestro.

5. Si aún no se ha logrado la convergencia global, entonces los esclavos solicitarán los valores en los nodos "fantasma" a cada una de las particiones adyacentes con la función MPI_Irecv$\left(j, \boldsymbol{G}^{ij}\right)$, con $j=1,\ldots,P$, $i\neq j$, al mismo tiempo enviara los valores de los nodos fantasma que soliciten las otras particiones con el la función MPI_Isend$\left(j, \boldsymbol{G}^{ji}\right)$, $j=1,\ldots,P$, $i\neq j$.

Con la función MPI_Wait esperaran a que todas las transacciones con las particiones adyacentes hayan concluido.



Esclavos $i=1,\ldots,P$

MPI_Irecv$\left(j, \boldsymbol{G}^{ij}\right)$

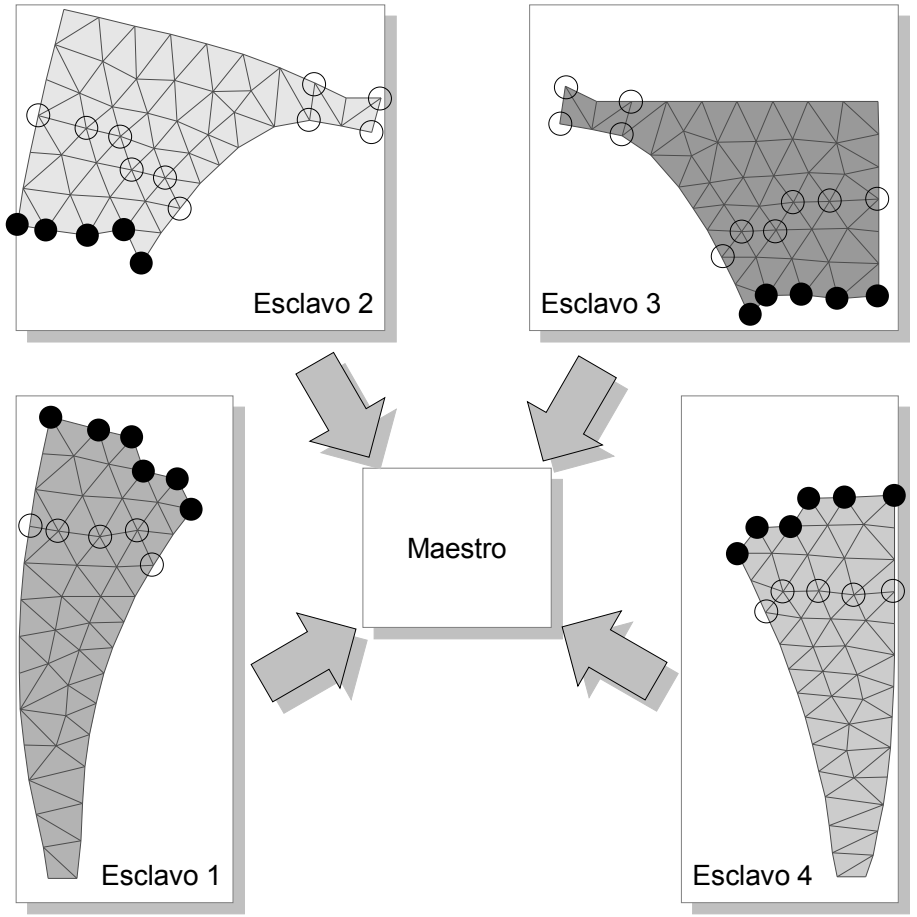
MPI_Isend$\left(j, \boldsymbol{G}^{ji}\right)$

MPI_Waitall$\left(\boldsymbol{G}^{ij}\right)$

MPI_Waitall$\left(\boldsymbol{G}^{ji}\right)$

Intercambio de valores en las fronteras artificiales con sus respectivos nodos fantasma.

6. Una vez que se logre la convergencia el nodo maestro entrará en un ciclo $i=1,\ldots,P$ y solicitará uno a uno a los esclavos los resultados de desplazamiento $\boldsymbol{u}^i$, deformación $\boldsymbol{\varepsilon}^i$, y esfuerzos $\boldsymbol{\sigma}^i$ de cada partición.



Maestro

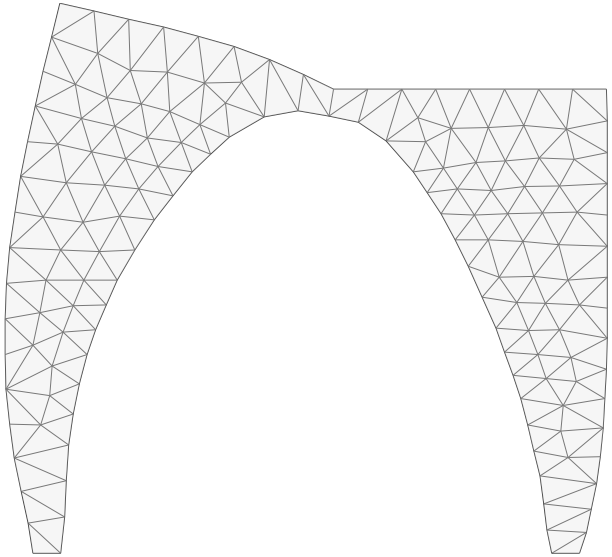
$$\text{MPI\_Recv}\left(i,\boldsymbol{u}^i\right)$$
$$\text{MPI\_Recv}\left(i,\boldsymbol{\varepsilon}^i\right)$$
$$\text{MPI\_Recv}\left(i,\boldsymbol{\sigma}^i\right)$$

Esclavos $i=1,\ldots,P$

$$\text{MPI\_Send}\left(0,\boldsymbol{u}^i\right)$$
$$\text{MPI\_Send}\left(0,\boldsymbol{\varepsilon}^i\right)$$
$$\text{MPI\_Send}\left(0,\boldsymbol{\sigma}^i\right)$$

Envio de resultados locales al nodo maestro.

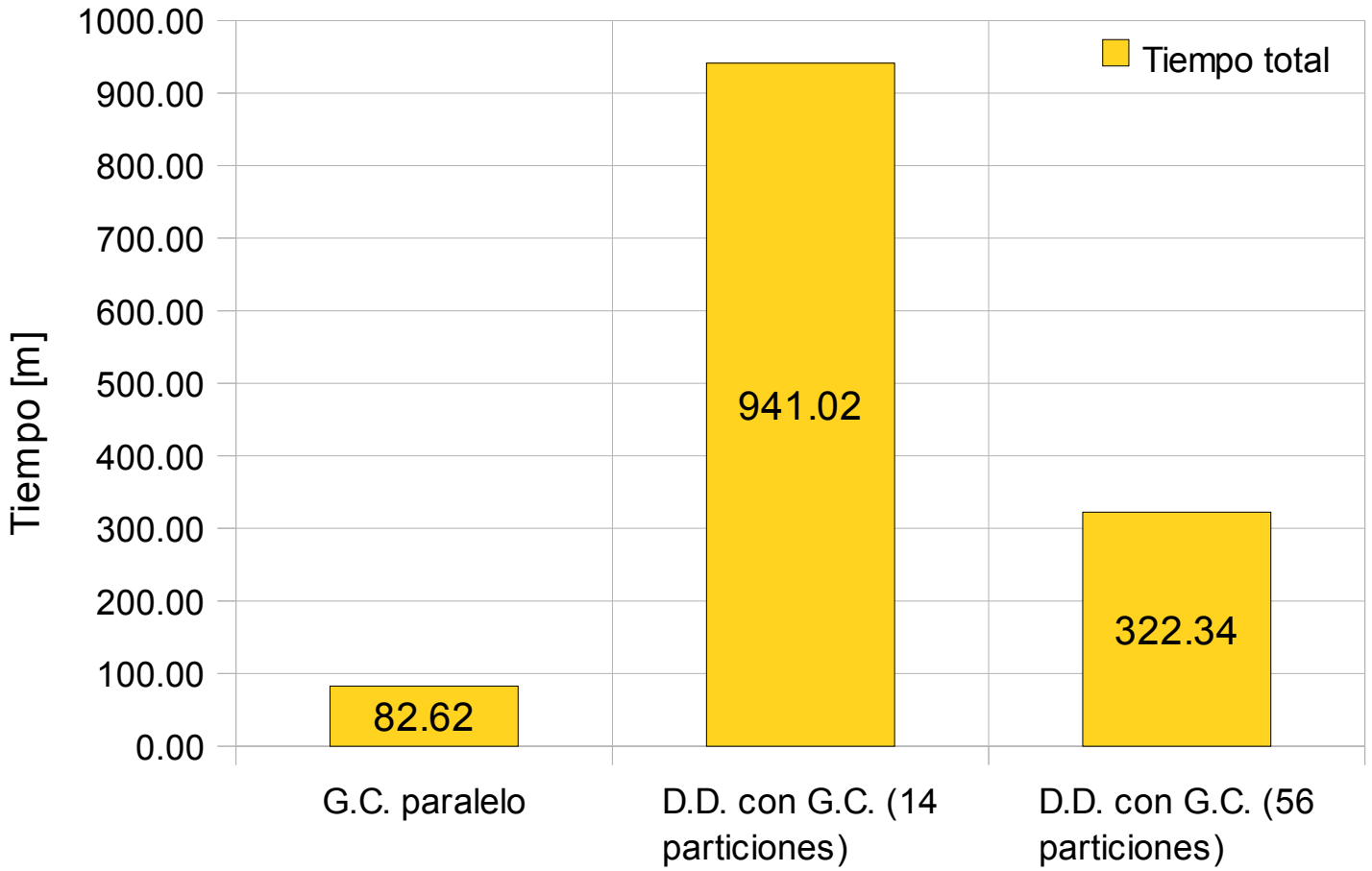
7. El nodo maestro generará una solución global conjuntando los resultados de todas las particiones. Los nodos salen del esquema MPI llamando la función MPI_Finalize.



Consolidación de un resultado global.

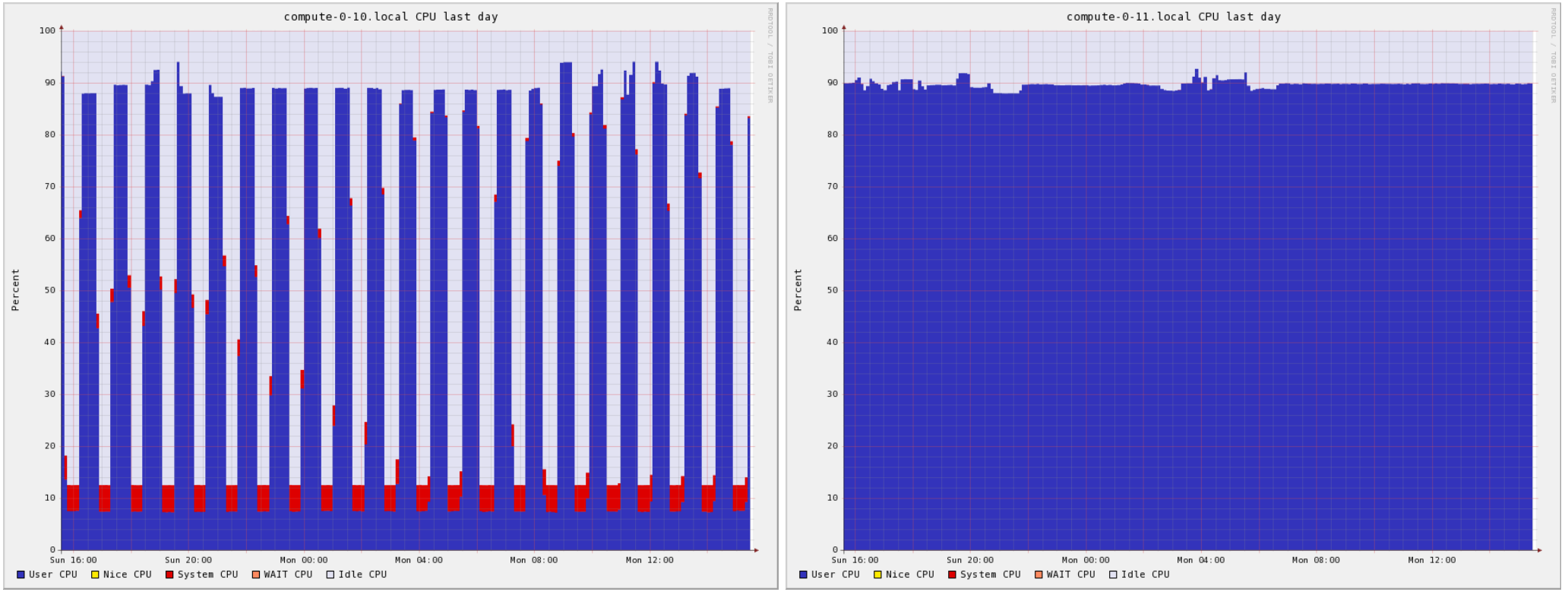
Método de Schwarz con gradiente conjugado

En las siguientes pruebas se siguió utilizando el "Cluster 2" del CIMAT, utilizamos las 14 computadoras del cluster, cada una con cuatro procesadores, lo que nos da un total de 56 cores.



Comparación de estrategias usando gradiente conjugado.

El problema que encontramos con este esquema es que el tiempo que requiere el gradiente conjugado para converger es muy diferente en cada una de las particiones.

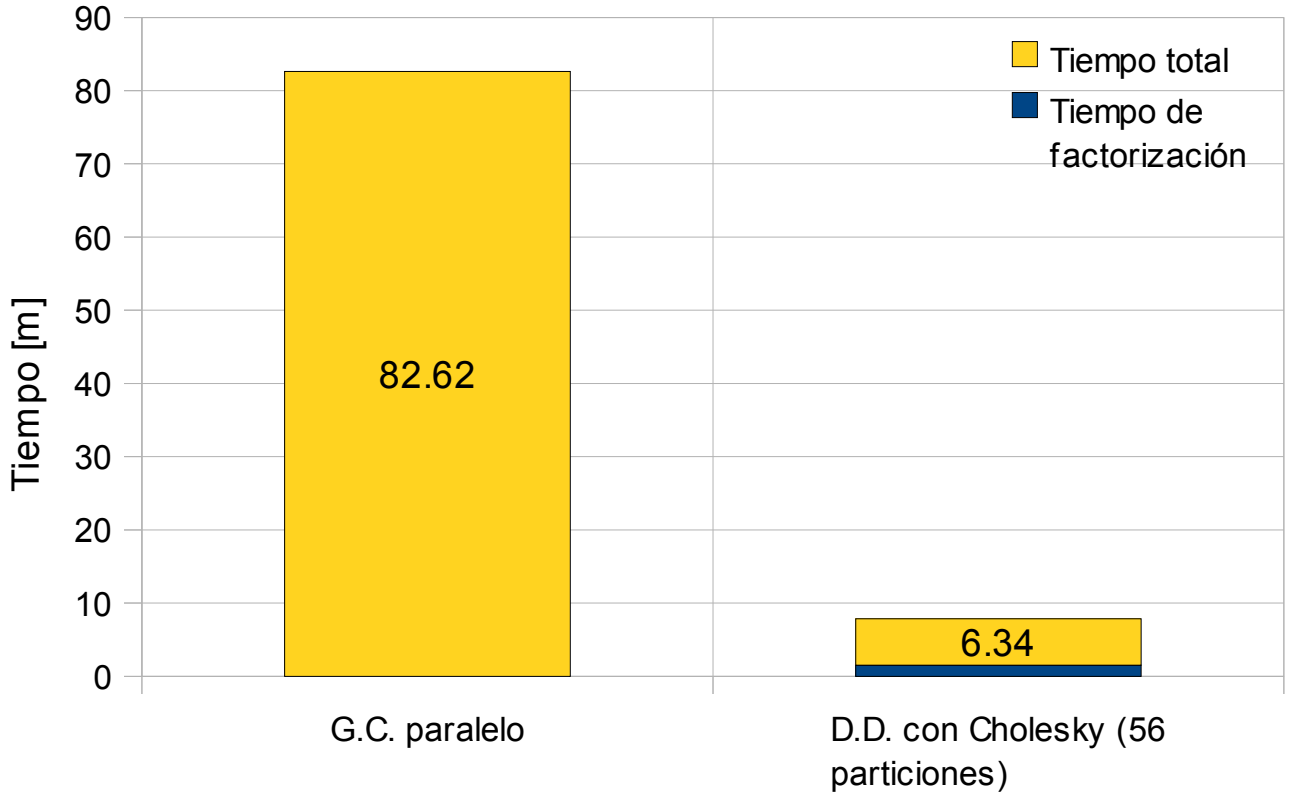


Diferencia de carga en los nodos más eficiente (izquierda) y menos eficiente (derecha)

En el nodo más eficiente se ve que después de terminar de resolver el sistema tiene periodos de baja actividad, durante este tiempo este nodo está esperando que los otros nodos terminen. En comparación, el nodo menos eficiente tarda más en resolver el sistema de ecuaciones, al ser el más lento no muestra periodos de espera, en la gráfica se aprecia que está trabajando continuamente. El nodo menos eficiente alenta a todos los nodos del cluster.

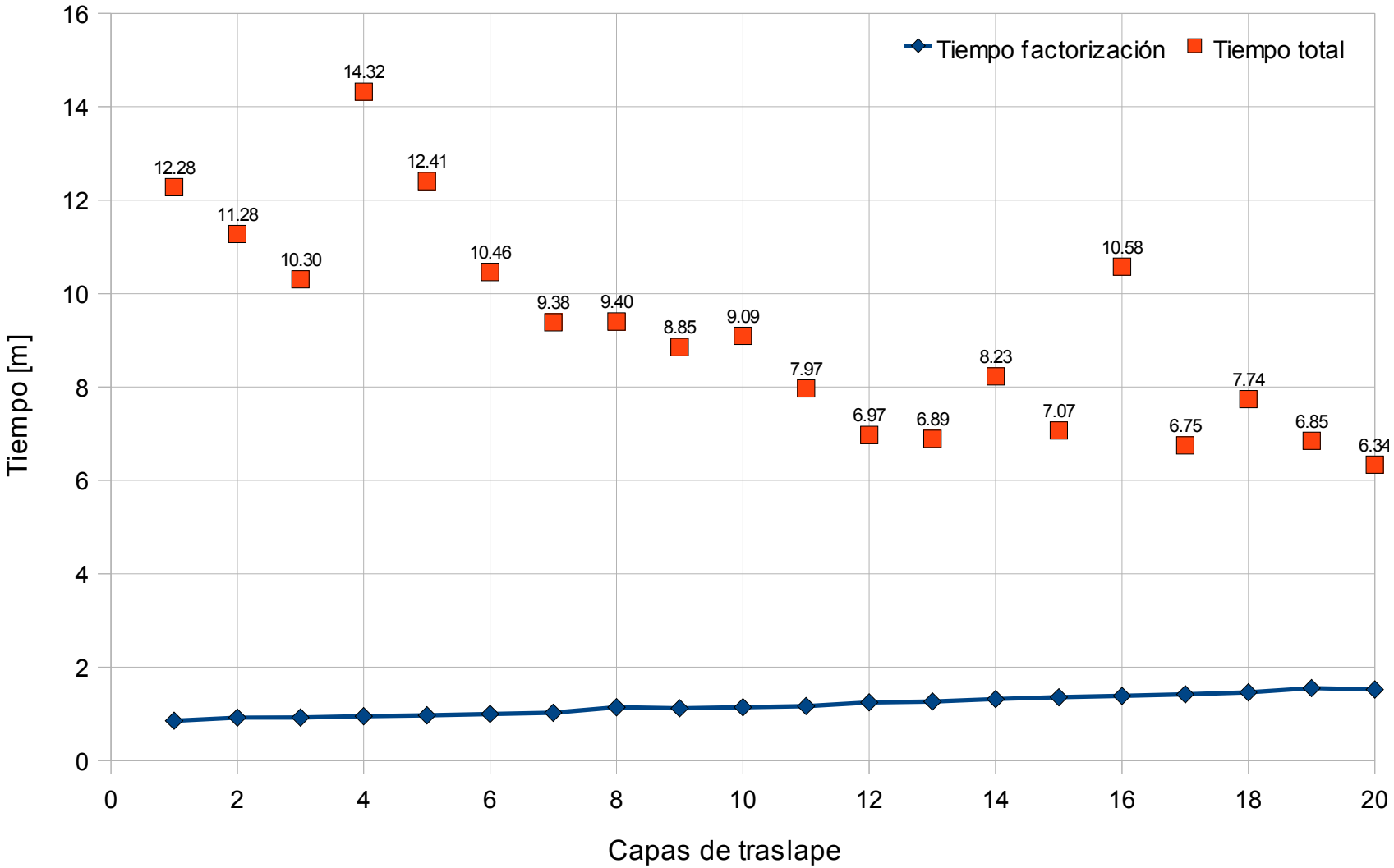
Método de Schwarz con Cholesky

Entre más pequeño sea el sistema de ecuaciones más eficiente será la resolución por factorización Cholesky, es por eso que en vez de utilizar 14 particiones, como en el caso con gradiente conjugado paralelizado, utilizaremos 56 particiones (una por cada procesador del cluster) con 20 capas de traslape.



Comparación de estrategias usando Cholesky.

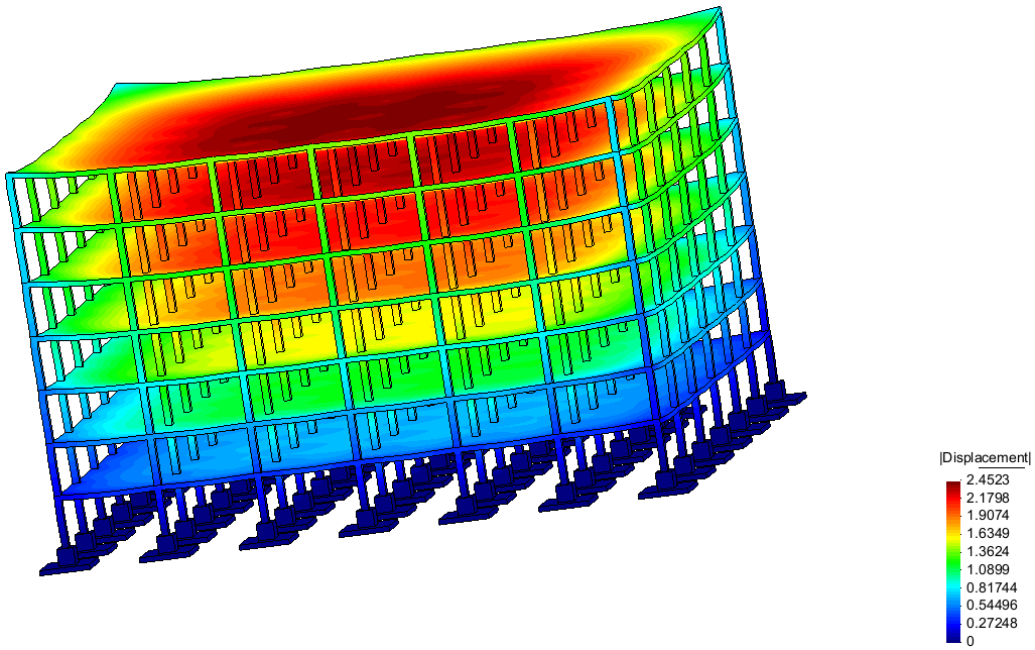
El efecto de las capas de traslape



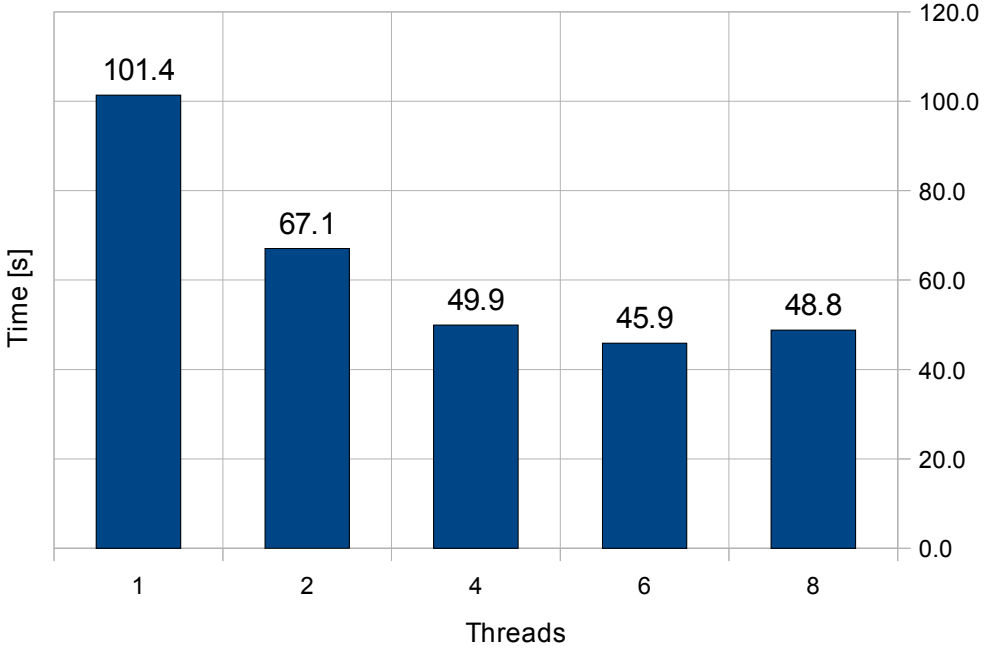
Como se ve hay una tendencia de que al aumentar el número de capas de traslape se aumenta el tiempo necesario para factorizar, pero se reduce el tiempo total para la resolución del problema.

Ejemplo con Cholesky en paralelo

Problem:	Edificio
Dimension:	3
Elements:	264,250
Nodes:	326,228
Variables:	978,684
nnz(A):	69,255,522
Global tolerance	1e-4
Partitions	27
Overlapping	3
Solver	Parallel cholesky



Number of threads	Total time [s]	Total iterations	Iteration time [s]
8	48.8	204	0.114
6	45.9	204	0.095
4	49.9	204	0.092
2	67.1	204	0.095
1	101.4	204	0.096



La memoria total: 13,438,244,806 [bytes]

Memoria nodo maestro: 308,703,548 [bytes]

Memoria en nodos esclavo: 486,279,305 [bytes]

¿Preguntas?

miguelvargas@cimat.mx
http://www.cimat.mx/~miguelvargas

Bibliografía:

[Smit96]B. F. Smith, P. E. Bjorstad, W. D. Gropp. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge Univeristy Press, 1996.

[Tose05]A. Toselli, O. Widlund. Domain Decomposition Methods - Algorithms and Theory. Springer, 2005.