Capítulo 3

Influencia del modelo de comportamiento en el proceso de mapping

En este capítulo mostramos la aplicabilidad de la información temporal incluida en el TTIG para resolver distintos aspectos del problema del mapping. Por un lado se estudia cómo la inclusión del grado de paralelismo en el modelo de comportamiento, propicia la definición de políticas más eficientes y a la vez más estables. Por otro lado, se propone un mecanismo para resolver una estimación de la cota mínima y máxima del número de procesadores para realizar la asignación, de tal forma que se explote la máxima concurrencia de la aplicación, sin utilizar más procesadores de los necesarios.

3.1 Bondad y sensibilidad de la asignación

El conocimiento del grado de paralelismo proporciona determinadas facilidades para la obtención de asignaciones más eficientes. A continuación se estudian mediante ejemplos los siguientes aspectos relacionados con el proceso de asignación: (a) influencia del grado de paralelismo en la asignación de las tareas adyacentes, (b) sensibilidad de las asignaciones frente a las variaciones en los parámetros de entrada y, (c) bondad de las asignaciones obtenidas evaluando las tareas con dependencias fuertes (es decir, las que exhiben un grado de

paralelismo bajo). Para los ejemplos desarrollados se considera la asignación a dos procesadores {P0,P1}. Dichos ejemplos se han escogido intencionadamente sencillos, con el objetivo de hacer hincapié en el aspecto que se quiere mostrar.

3.1.1 Influencia del grado de paralelismo

Consideremos un programa paralelo con tres tareas, cuyo comportamiento temporal corresponde al grafo TFG de la Figura 3.1(a). A partir de este TFG, se pueden extraer los dos modelos TIG y TTIG mostrados en la Figura 3.1(b) y la Figura 3.1(c) respectivamente.

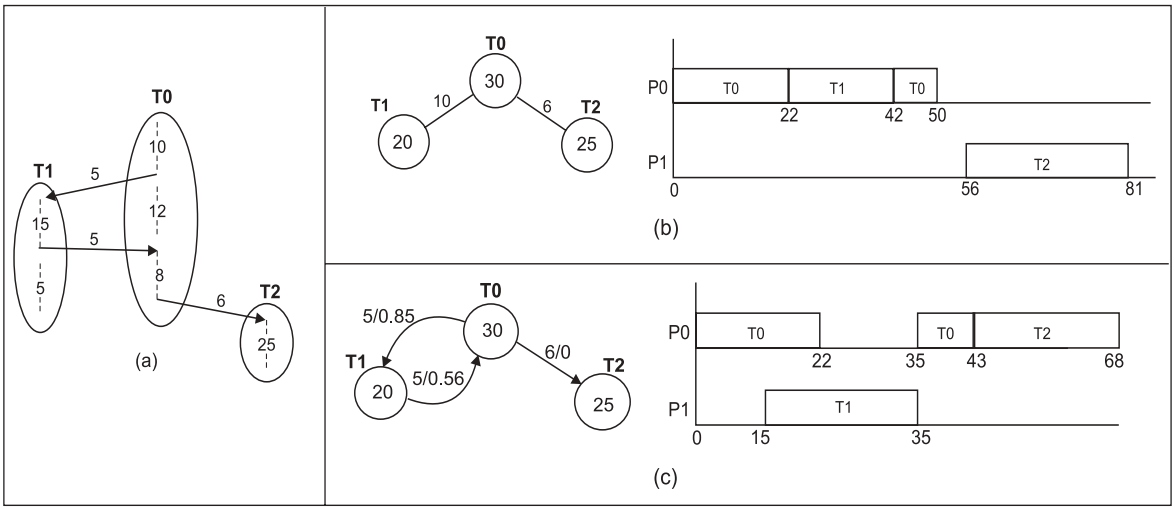

Figura 3.1: (a) Grafo TFG de un programa. (b) Grafo TIG. (c) Grafo TTIG.

Partiendo de la información del modelo TIG para el ejemplo considerado, la asignación que mejor equilibra el cómputo y minimiza las comunicaciones entre procesadores, es la que asigna las tareas T0 y T1 al mismo procesador y T2 al otro procesador. Como puede verse en la simulación asociada a esta asignación, se da una escasa utilización de los procesadores, puesto que las tareas no pueden paralelizar en ningún momento su ejecución.

Mediante el análisis del grafo TTIG para este mismo programa, una estrategia de mapping decidirá fácilmente asignar las tareas T0 y T2 al mismo procesador, puesto que su grado de paralelismo es 0, y esto implica directa-

mente que deberán ser ejecutadas de forma secuencial. La tarea T1 podrá ser asignada al otro procesador ya que su grado de paralelismo con T0 muestra una capacidad media de ejecución paralela que puede ser explotada. En la Figura 3.1(c) puede verse la simulación correspondiente a esta ejecución, con la consecuente mejora de tiempo y aprovechamiento de los procesadores.

3.1.2 Sensibilidad de la asignación a las variaciones en la estimación de los costes asociados al modelo

Como hemos visto en la Sección 2.2, cuando se modela el comportamiento del programa de forma estática, los valores de los parámetros asociados al modelo se obtienen mediante determinadas técnicas que permiten hacer una estimación de los mismos. Cuando el comportamiento del programa es predecible, estos parámetros son representativos y permitirán desarrollar técnicas de mapping eficientes basadas en su valor. De todos modos, cuando la estimación precisa de dichos valores es difícil, pueden existir ciertas variaciones entre los valores estimados y los que se dan en tiempo de ejecución.

En el caso de usar el modelo TIG, las asignaciones obtenidas son fuertemente dependientes de los valores de cómputo y comunicación estimados. Pequeñas variaciones en los valores estimados de estos parámetros pueden conducir (según la estructura del grafo) a asignaciones completamente distintas. Para verlo, consideremos como ejemplo representativo de este caso el programa paralelo cuyo comportamiento se modela con el grafo TFG de la Figura 3.2(a), en el cual el volumen de comunicación a transmitir entre las tareas T0 y T2 puede ser estimado con una pequeña variación entre 9 y 12. Los modelos TIG y TTIG que se derivan de este grafo se muestran en la Figura 3.2(b) y 3.2(c) respectivamente.

La asignación del programa basada en el modelo TIG, considerando que el volumen de comunicación entre T0 y T2 es 12, será P0:(T0,T2) y P1:(T1). De esta forma se minimizan las comunicaciones. La asignación que se obtendrá a partir del análisis del grafo TTIG será ésta misma, ya que el grado de paralelismo entre T0 y T2 es mucho menor que el existente entre T0 y T1.

Si el volumen de comunicación entre T0 y T2 se estima en 9 en lugar de

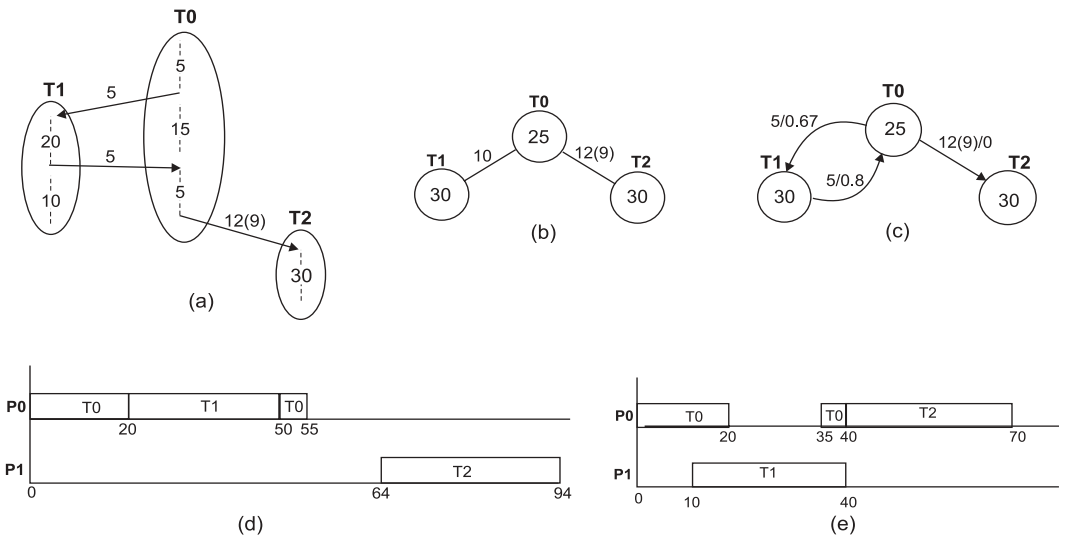

Figura 3.2: (a) Grafo TFG de un programa. (b) TIG. (c) TTIG. (d) Simulación de la asignación basada en TIG. (e) Simulación de la asignación basada en TTIG.

12, la asignación basada en el modelo TIG va a ser completamente diferente: P0:(T0,T1) y P1:(T2). Con esta asignación se minimizan las comunicaciones entre los dos procesadores para este caso. La Figura 3.2(d) muestra la simulación para esta asignación. En cambio, esta pequeña variación en la estimación de las comunicaciones no va a tener incidencia en la asignación que se obtenga a partir del modelo TTIG, ya que los grados de paralelismo no cambian y continuarán mostrando que existe una dependencia fuerte entre T0 y T2 y la asignación final va a continuar siendo la misma. La Figura 3.2(e) corresponde a la simulación de esta asignación, en la que se observa un mejor tiempo de ejecución final.

Vemos pues con este sencillo ejemplo, que el modelo TTIG es en sí mismo menos sensitivo que el TIG a pequeñas variaciones en la estimación de los pesos asociados al cómputo y a las comunicaciones. Esto permitirá por un lado, definir políticas de mapping más eficientes y por otro, simplificará el proceso y las herramientas de evaluación de estos pesos.

3.1.3 Bondad de las asignaciones basadas en la evaluación de las tareas más dependientes

El grado de paralelismo incluido en el modelo TTIG da una información clara respecto al tipo de dependencias que existen entre las tareas adyacentes. Teniendo en cuenta únicamente las tareas que son más dependientes es posible generar una asignación con ciertas garantías que su tiempo de ejecución final sea aceptable. En el presente apartado mostramos esta idea mediante un ejemplo, aún teniendo presente que el uso de una técnica de mapping más refinada, que contemple todas las dependencias del grafo, proporcionará de forma más fiable un mejor tiempo de ejecución final.

Dado un grafo TTIG, consideramos que dos tareas adyacentes tienen una dependencia fuerte si produce mejor tiempo su ejecución juntas al mismo procesador que separadas en distintos procesadores (es decir, se trata de tareas adyacentes con un grado de paralelismo bajo, o un volumen de comunicación elevado, o ambas cosas).

Podemos comprobar que en general, de entre las asignaciones que equilibran el cómputo de la aplicación, aquellas que *respetan* las dependencias fuertes producen un tiempo final de ejecución eficiente.

Estudiemos a título de ejemplo las asignaciones de un programa con cinco tareas con igual tiempo de cómputo, cuyo Grafo de Flujo Temporal se muestra en la Figura 3.3(a). El modelo TTIG asociado a este programa será el de la Figura 3.3(b), donde la diferencia entre las dependencias viene determinada por los valores del grado de paralelismo, puesto que el volumen de comunicación es insignificante en todos los casos.

Mediante el análisis de estos grados de paralelismo podemos determinar que las dependencias fuertes de este grafo son las que existen entre los pares de tareas (T0,T3) y (T1,T4), ya que su grado de paralelismo es cero, mientras que las restantes tareas tienen un paralelismo potencial alto o medio. A raíz de esto, es obvio pensar que una buena asignación de esta aplicación será aquella que junta en el mismo procesador los pares de tareas con dependencia fuerte, mientras que reparte el cómputo de manera equilibrada con las restantes.

Si se simula la ejecución de este grafo TFG para las 25 asignaciones posibles

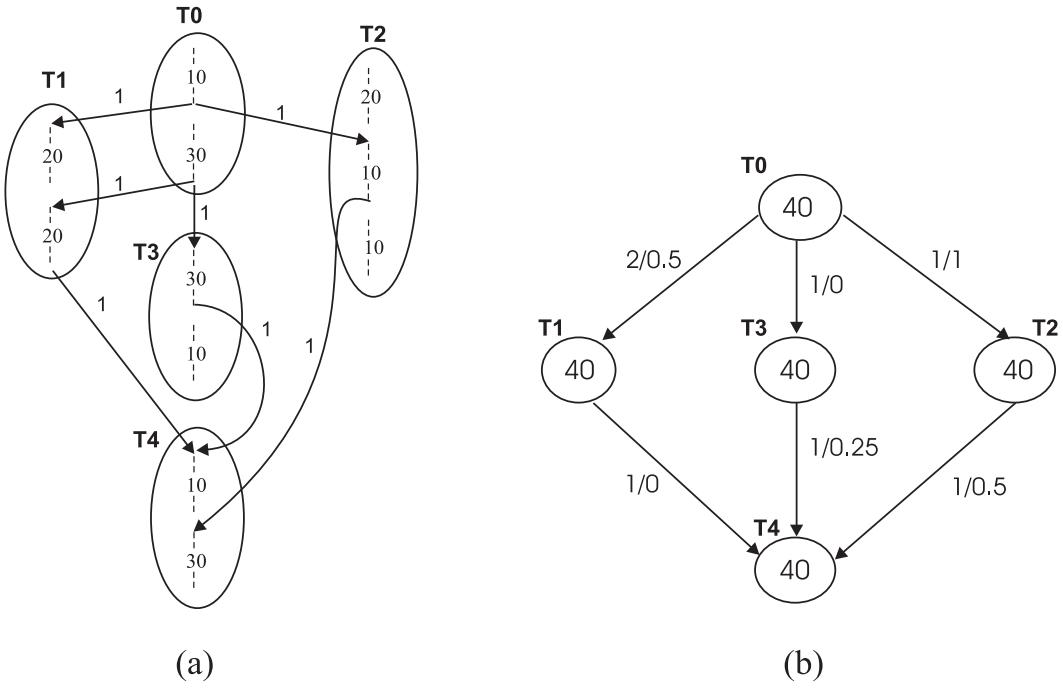

Figura 3.3: (a) Grafo TFG de un programa. (b) Modelo TTIG.

que se pueden obtener usando tres procesadores {P0,P1,P2}, el valor del tiempo de ejecución oscila desde 111 unidades de tiempo, que es el tiempo óptimo, hasta 171 para la peor asignación. Puesto que todas las tareas tienen el mismo tiempo de ejecución, las asignaciones que equilibran mejor el cómputo son aquellas que tienen dos procesadores con dos tareas y un procesador con una. En la Tabla 3.1 se muestran las 15 asignaciones que cumplen este requisito de entre las 25 posibles, donde para cada asignación figuran entre paréntesis las tareas asignadas a cada uno de los procesadores P0, P1 y P2 respectivamente. Para cada asignación se ha puesto el tiempo de ejecución que le corresponde obtenido mediante simulación, y están indicadas con una x las asignaciones que juntan las tareas de cada una de las dependencias fuertes (T0,T3) y (T1,T4) indicadas en cada columna.

Se puede observar en la tabla que dos de las asignaciones que contemplan las dependencias dominantes dan el tiempo óptimo. De las tres asignaciones restantes que incorporan una de las dos dependencias dominantes, dos de ellas dan una desviación de un 0.9% y un 9% respecto al óptimo, mientras que sólo

Tabla 3.1: Asignaciones que equilibran el cómputo.

Asignación	tiempo	(T0,T3)	(T1,T4)
(2)(0,3)(1,4)	111	x	x
(1)(0,3)(2,4)	111	x	
(3)(0,2)(1,4)	112		x
(2)(0,1)(3,4)	121		
(3)(0,1)(2,4)	121		
(1)(2,3)(0,4)	121		
(4)(0,1)(2,3)	121		
(1)(0,2)(3,4)	121		
(0)(2,3)(1,4)	121		x
(3)(1,2)(0,4)	131		
(4)(1,2)(0,3)	131	x	
(0)(1,2)(3,4)	131		
(2)(1,3)(0,4)	132		
(4)(0,2)(1,3)	132		
(0)(1,3)(2,4)	132		

una proporciona una desviación del 18%.

Como se ha mostrado en el Capítulo 2, la construcción del modelo TTIG comporta una complejidad adicional respecto alguno de los modelos clásicos, pero vemos por otra parte que únicamente a partir del análisis de las dependencias reflejadas en el grafo, es relativamente sencillo escoger una buena asignación, reduciendo sustancialmente el tiempo requerido en el proceso de asignación.

3.2 Utilización del modelo de comportamiento para determinar una cota sobre el número de procesadores

En el momento de realizar el proceso de mapping, un punto importante a decidir es el número de procesadores a utilizar para realizar la asignación de las tareas del programa. A partir del modelo de comportamiento puede establecerse un intervalo respecto al número de procesadores, que indica la cota mínima y máxima respectivamente, cuyos valores se definen de la siguiente

forma:

- *Cota mínima.* Indica el mínimo número de procesadores que se precisan para ejecutar las tareas de la aplicación en el mínimo tiempo alcanzable. Es decir, utilizando un número de procesadores igual al de la cota mínima, siempre será posible encontrar una asignación de tareas que ejecute la aplicación en un tiempo óptimo.

- *Cota máxima.* Indica el máximo número de procesadores que será necesario para explotar la máxima capacidad de paralelismo de la aplicación. Es decir, el uso de un número de procesadores mayor que el indicado por la cota máxima no redundará en un menor tiempo de ejecución, puesto que la aplicación no posee mayor capacidad de concurrencia.

Los valores de cota están basados en el conocimiento del tiempo de ejecución de la aplicación. Debido a ello, éstos únicamente pueden obtenerse a partir de modelos de la aplicación que contengan información temporal de la misma.

Clásicamente este problema ha sido estudiado por diversos autores para Grafos de Precedencia de Tareas (TPG), utilizando distintas restricciones:

- Cotas de Hu [Hu61]. Calcula los valores de cota para grafos cuyas tareas tienen tiempo de cómputo uniforme y topología en árbol.

- Cotas de Ramamoorthy [RCG72]. Extiende el resultado de Hu para topologías arbitrarias de grafos de precedencia.

- Cotas de Fernandez y Bussel [FB73]. Proporciona un valor de cota de procesadores para grafos cuyas tareas pueden tener un tiempo de cómputo arbitrario con valor entero. La topología puede ser también arbitraria.

En todos los casos citados se consideran grafos TPG sin costes de comunicación, con lo cual los valores de cota que se obtienen corresponden a la estructura de dependencias entre tareas, sin contemplar los posibles retardos de comunicación.

La cotas de Fernandez y Bussel son, en sí mismas, una extensión de las dos restantes mencionadas. Estas se basan en la consideración de todos los posibles intervalos enteros de ejecución que forman el camino crítico del grafo. Puesto que el camino crítico es el camino más largo desde un nodo inicial hasta un nodo final, para grafos TPG sin volúmenes de comunicación el tiempo del camino crítico, que denominamos t_{cp}, corresponde al mínimo tiempo en que pueden ser ejecutadas todas las tareas del grafo.

El Grafo de Flujo Temporal (TFG) propuesto en este trabajo como modelo de comportamiento compactado para las aplicaciones con paso de mensajes, constituye en sí mismo un grafo de precedencias entre las fases de cómputo especificadas en el mismo. Debido a ello, hemos adaptado el método de Fernandez y Bussel, al grafo TFG sin volúmenes de comunicación, aplicando la restricción que todas las fases de cómputo que pertenecen a una misma tarea han de ser ejecutadas en un mismo procesador (ya que las fases de cómputo son únicamente porciones de cómputo secuencial que no tienen entidad de tareas en sí mismas).

El TFG sin comunicaciones forma un grafo dirigido acíclico $G(I, A)$, donde $I = \{I0, .., In\}$ constituye el conjunto de fases de cómputo que forman las distintas tareas del grafo, y $A = \{(Ii, Ij)\, i, j \in I\}$ es el conjunto de arcos existentes entre estas fases de cómputo. El grafo $G(I, A)$, define un orden parcial de ejecución, $(I, <)$, entre las fases de cómputo del grafo. A partir de este orden parcial se determinan los límites inferior y superior del número de procesadores necesario para ejecutar las tareas del grafo de tal forma que el tiempo global de ejecución sea t_{cp}.

En la Figura 3.4 se muestra un ejemplo de grafo TFG sin volúmenes de comunicación, donde las fases de cómputo están indicadas con una etiqueta Ii. Observando el grafo vemos a modo de ejemplo que la fase de cómputo $I9$ no puede empezar hasta que sus predecesoras $I5$ y $I2$ hayan finalizado. La fase de cómputo $I10$ deberá esperar a que $I7$ y $I9$ hayan acabado.

Considerando el grafo dirigido acíclico que forman las fases de cómputo de este grafo, se obtiene un valor $t_{cp} = 110$, que corresponde al tiempo del camino crítico formado por $I0, I1, I2, I9, I10$. Dicho grafo será el que se utilice como ejemplo para el cálculo de las cotas, donde para cada fase de cómputo Ii se

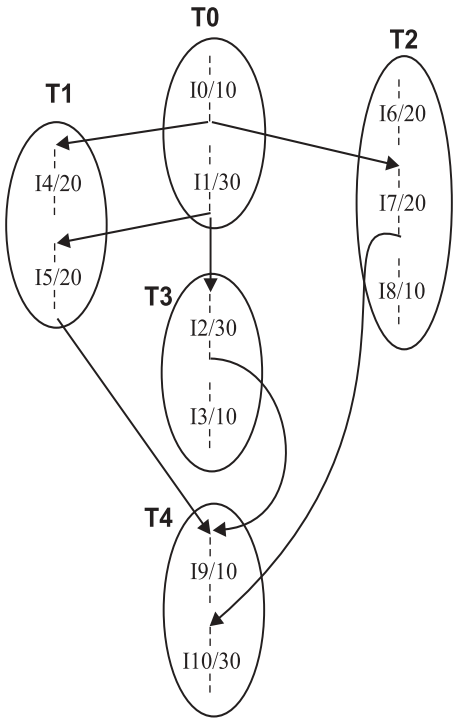

Figura 3.4: Grafo de Flujo Temporal sin volúmenes de comunicación.

calculan los siguientes parámetros:

- *Mínimo tiempo de finalización ($\underline{\tau}_i$).* Corresponde al mínimo valor de tiempo en que la fase de cómputo Ii puede haber finalizado, y se define como:

$$\underline{\tau}_i = max_k \sum_{u \in \pi_k} t_u \tag{3.1}$$

  donde π_k es el camino k desde un vértice de entrada del grafo hasta la fase de cómputo Ii del grafo.

- *Máximo tiempo de finalización ($\bar{\tau}_i$).* Indica el máximo valor de tiempo en que puede ser postpuesta la finalización de una fase de cómputo Ii, sin que se incremente el tiempo del camino crítico del grafo. Se define como:

$$\bar{\tau}_i = min_k[t_{cp} - \sum_{u \in \hat{\pi}_k} t_u] \tag{3.2}$$

donde:

t_{cp} es el tiempo del camino crítico.

$\hat{\pi}_k$ corresponde al camino k desde un vértice de salida hasta la fase de cómputo Ii.

- *Intervalo de finalización.* Indica el rango de valores de tiempo en los cuales una fase de cómputo Ii tiene que haber acabado su ejecución para que el grafo completo pueda finalizar en un tiempo igual a t_{cp}. El intervalo de finalización estará formado por los dos valores calculados $[\underline{\tau}_i, \bar{\tau}_i]$.

A continuación se expone el procedimiento para obtener los valores de cota mínima y máxima de procesadores a partir de los valores que tienen los intervalos de finalización de las fases de cómputo.

3.2.1 Cálculo del valor de cota mínima

Dada la simulación del grafo TFG formada por un conjunto de intervalos de ejecución $[ti, tj] \in [0, t_{cp}]$, la mínima cota de procesadores, m_L, se calcula como:

$$m_L = \lceil max_{[t_i,t_j]}[\frac{1}{n_{int}}|\underline{F} \cap \bar{F}|]\rceil \tag{3.3}$$

donde:

n_{int} indica el número de intervalos de tiempo de ejecución que se está evaluando desde t_i a t_j.

$\underline{F}$ es el conjunto de las fases de cómputo cuyo mínimo tiempo de finalización está dentro del intervalo $[t_i, t_j]$.

$\bar{F}$ es el conjunto de fases de cómputo cuyo máximo tiempo de finalización está dentro del intervalo $[t_i, t_j]$.

$\underline{F} \cap \bar{F}$ indica las fases de cómputo (o partes de fases de cómputo), cuyo intervalo de finalización está dentro de $[t_i, t_j]$.

La metodología a seguir para calcular los parámetros de tiempo descritos, y la cota mínima de procesadores a partir de un grafo TFG será la que se indica en los siguientes pasos:

1. *Cálculo del mínimo tiempo de finalización de cada fase de cómputo.* Se obtiene a partir de la simulación del grafo TFG con una tarea por procesador. La Figura 3.5 muestra dicha simulación para las fases de cómputo del grafo de la Figura 3.4. A partir de ésta puede determinarse el valor de $\underline{\tau}_i$ correspondiente a la expresión 3.1 para cada fase de cómputo Ii. A título de ejemplo vemos que la fase de cómputo $I2$ finaliza en el tiempo igual a 70 , por lo tanto $\underline{\tau}_2 = 70$, para $I3$ se tiene $\underline{\tau}_3 = 80$.

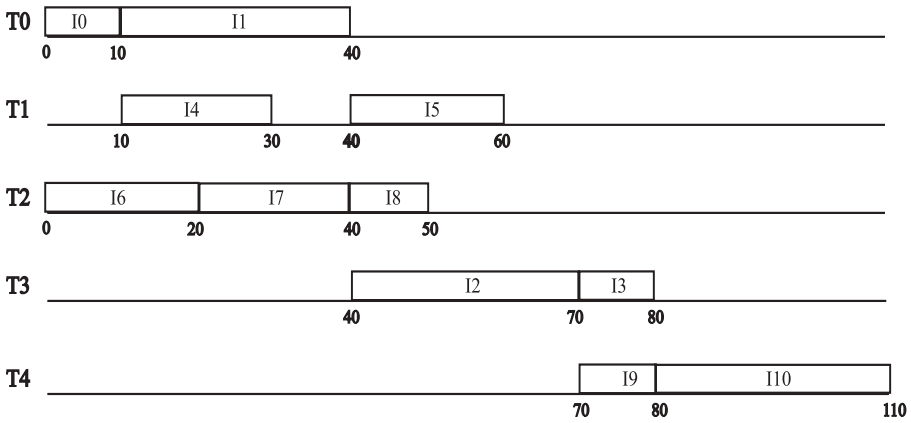

Figura 3.5: Mínimo tiempo de finalización.

2. *Cálculo del máximo tiempo de finalización de cada fase de cómputo.* A partir de la simulación del grafo TFG en orden inverso. Es decir, empezando por las fases de cómputo que constituyen nodos de salida del grafo y por el tiempo t_{cp}, se simula la ejecución del grafo yendo hacia atrás en el tiempo, y considerando los arcos en dirección inversa. La Figura 3.6 muestra dicha simulación, a partir de la cual se calcula el valor $\bar{\tau}_i$ de la expresión 3.2, para cada fase de cómputo Ii. Vemos como ejemplo que para este grafo TFG $\bar{\tau}_2 = 70$ y $\bar{\tau}_3 = 110$.

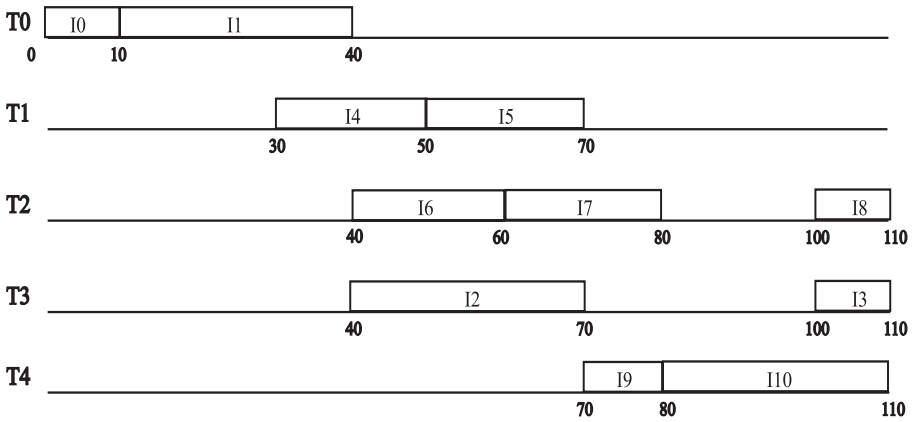

Figura 3.6: Máximo tiempo de finalización.

3. *Fases de cómputo de cada intervalo de finalización.* Para todas la combinaciones de intervalos de ejecución $[t_i, t_j]$, se calcula la intersección formada por las tareas cuyo intervalo de finalización (o parte de él) está dentro de $[t_i, t_j]$.

En la Tabla 3.2 se muestra el número de fases de cómputo que forman la intersección de $\underline{F} \cap \bar{F}$ para cada intervalo de ejecución y el valor de la mínima cota m_L calculada hasta el momento. Podemos observar por ejemplo en la tabla, que en el intervalo de ejecución [10,20] hay una fase de cómputo común que corresponde a $I1$, por lo tanto la cota mínima hasta este momento es $m_L = 1/1 = 1$. En cambio en el intervalo [50,60] las fases de cómputo comunes son $I2$ y $I5$, por lo tanto para este intervalo la cota pasa a ser $m_L = 2/1 = 2$. A partir de este momento el valor de m_L no se incrementa, ya que aunque haya más tareas comunes también se evaluan más intervalos de ejecución. Vemos por ejemplo que en el intervalo $[0, 50]$ hay 5 fases de cómputo comunes que corresponden a $I0, I1, I2, I4, I6$, y el valor de cota sería $m_L = 5/5 = 1$, pero m_L figura como 2 en la tabla puesto que se guarda el máximo valor de cota obtenido hasta el momento.

A partir del procedimiento descrito se ha obtenido el valor $m_L = 2$ como cota mínima del número de procesadores. Esto significa que este será necesario un mínimo de dos procesadores para ejecutar la aplicación correspondiente en el mínimo tiempo posible.

Tabla 3.2: Número de fases de cómputo comunes y valor de cota acumulado para los intervalos de ejecución.

n_{int}=1			n_{int}=2			n_{int}=3		
[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L
[0,10]	1	1	[0,20]	2	2	[0,30]	2	2
[10,20]	1	1	[10,30]	1	2	[10,40]	2	2
[20,30]	1	1	[20,40]	2	2	[20,50]	3	2
[30,40]	1	1	[30,50]	2	2	[30,60]	3	2
[40,50]	1	1	[40,60]	2	2	[40,70]	2	2
[50,60]	2	2	[50,70]	2	2	[50,80]	3	2
[60,70]	1	2	[60,80]	2	2	[60,110]	4	2
[70,80]	1	2	[70,110]	3	2			
[80,110]	1	2						

n_{int}=4			n_{int}=5			n_{int}=6		
[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L
[0,40]	3	2	[0,50]	5	2	[0,60]	6	2
[10,50]	4	2	[10,60]	5	2	[10,70]	6	2
[20,60]	4	2	[20,70]	5	2	[20,80]	6	2
[30,70]	4	2	[30,80]	5	2	[30,110]	8	2
[40,80]	3	2	[40,110]	6	2			
[50,110]	5	2						

n_{int}=7			n_{int}=8			n_{int}=9		
[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L	[ti,tj]	$\cap$	m_L
[0,70]	7	2	[0,80]	8	2	[0,110]	11	2
[10,80]	7	2	[10,110]	10	2			
[20,110]	9	2						

3.2.2 Cálculo del valor de cota máxima

A partir de método propuesto por Fernanadez y Bussel puede determinarse también el valor de cota máxima del número de procesadores, que denotamos como m_U. Dicho valor corresponde al máximo número de fases de cómputo que pueden estar ejecutándose simultaneamente, y se calcula mediante la siguiente expresión:

$$m_U = min[max|\underline{F}|, max|\bar{F}|] \tag{3.4}$$

donde:

$max|\underline{F}|$ corresponde al máximo número de fases de cómputo cuyo mínimo tiempo de finalización está dentro del mismo intervalo de ejecución $[ti, tj]$.

$max|\bar{F}|$ corresponde al máximo número de fases de cómputo cuyo máximo tiempo de finalización está dentro del mismo intervalo de ejecución $[ti, tj]$.

La metodología para calcular el valor de m_U consta sencillamente de los siguientes pasos:

1. Cálculo del máximo número de fases de cómputo cuyo mínimo tiempo de finalización está dentro de cada intervalo $[ti, tj]$. Este se calculará a partir de la simulación del grafo TFG mostrada en la Figura 3.5.

2. Cálculo del máximo número de fases de cómputo cuyo máximo tiempo de finalización está dentro de cada intervalo $[ti, tj]$. Este se calculará a partir de la simulación del grafo TFG invertido de la Figura 3.6.

Para el grafo de la Figura 3.4, la Tabla 3.3 muestra para cada intervalo de cómputo los valores de $|\underline{F}|$ y $|\bar{F}|$, obtenidos a partir de los intervalos de cómputo de las trazas de simulación de las Figuras 3.5 y 3.6 respectivamente.

Vemos por ejemplo en la tabla que el valor de $|\underline{F}|$ en el intervalo $[60, 70]$ es de 1, que corresponde a la ejecución de la fase de cómputo $I2$, en cambio para el mismo intervalo el valor de $|\bar{F}|$ es de 3, que corresponde a la ejecución de las fases de cómputo $I2, I5$ y $I7$ dentro de este intervalo. Tanto para los intervalos de mínimo como de máximo tiempo de finalización se obtiene un máximo de

Tabla 3.3: Número de tareas simultáneas en cada intervalo de ejecución.

Mínimo tiempo finaliz.		Máximo tiempo finaliz.	
$[ti, tj]$	$\lvert\underline{F}\rvert$	$[ti, tj]$	$\lvert F\rvert$
$[0, 10]$	2	$[0, 20]$	1
$[10, 20]$	3	$[10, 30]$	1
$[20, 30]$	3	$[30, 40]$	2
$[30, 40]$	2	$[40, 50]$	3
$[40, 50]$	3	$[50, 60]$	3
$[50, 60]$	2	$[60, 70]$	3
$[60, 70]$	1	$[70, 80]$	2
$[70, 80]$	2	$[80, 100]$	1
$[80, 110]$	1	$[100, 110]$	3
$max\lvert\underline{F}\rvert = 3$		$max\lvert\bar{F}\rvert = 3$	
$m_U = min(3, 3) = 3$			

tres fases de cómputo ejecutándose simultaneamente, por lo tanto el valor final de cota máxima según la expresión 3.4 será $m_U = 3$. Esto significa que para este ejemplo no serían necesarios más de tres procesadores para explotar el paralelismo útil de sus tareas.

En el Capítulo 5 se valora la adecuación de los valores calculados de cota de procesadores, comparando con el número de procesadores que proporciona la asignación óptima. Para dar una idea de como estas cotas están de lejos con respecto a la realidad, se analizará el grado de utilización de los procesadores, admitiendo que son cotas muy teóricas si los procesadores están inactivos la mayor parte del tiempo o por el contrario se aproximan a la realidad si son utilizados fuertemente.