Capítulo 2

El modelo TTIG (Temporal Task Interaction Graph)

Para modelar el comportamiento de las aplicaciones paralelas, con el propósito del mapping, en este trabajo se propone el nuevo modelo de Grafo de Interacción de Tareas Temporal (TTIG: Temporal Task Interaction Graph) [RRS+00b]. Con respecto a los modelo clásicos el grafo TTIG incluye un nuevo parámetro denominado grado de paralelismo. Dicho parámetro es indicativo de la concurrencia potencial que las tareas adyacentes pueden asumir durante su ejecución en función de sus dependencias. Esta información permite la definición de nuevas políticas de mapping más eficientes.

El modelo TTIG puede ser deducido estáticamente mediante mecanismos automáticos, a partir de los programas con paso de mensajes, que pueden tener un patrón de interacción de tareas arbitrario.

2.1 Definición del modelo TTIG

Definimos el modelo TTIG como un grafo dirigido $G(V, E)$, junto con tres parámetros, donde:

- $V = \{T0, T1, .., Tn-1\}$ es el conjunto de nodos, donde cada nodo representa a una tarea. La tarea es la unidad indivisible de cómputo que se asigna a un procesador.

- E es el conjunto de arcos (Ti, Tj), para todo i,j tal que existe una comunicación de Ti a Tj.

- $w : V \rightarrow N$ asocia un tiempo de cómputo, $w(Ti)$, a cada tarea.

- $c : E \rightarrow N$ asocia un volumen de comunicación a los arcos, donde $c(Ti, Tj)$ es la cantidad global de datos que han de ser transferidos de Ti a Tj.

- $p : E \rightarrow [0, 1]$ será el grado de paralelismo. Este es un índice normalizado, denotado como $p(Ti, Tj)$, asociado a cada arco (Ti, Tj) del grafo.

  Para dos tareas Ti y Tj que se comunican de Ti a Tj, el grado de paralelismo se define como el máximo porcentaje del tiempo de cómputo de Tj que puede ser realizado en paralelo con Ti, teniendo en cuenta sus dependencias mutuas provocadas por las comunicaciones existentes entre ambas tareas, y sin contemplar el coste de comunicación asociado a las mismas. De esta forma, el grado de paralelismo constituye un valor independiente de la cantidad de datos a transmitir.

El nuevo parámetro grado de paralelismo introducido en el TTIG, hace que éste constituya una propuesta unificada de los dos modelos clásicos TPG y TIG vistos en los apartados 1.2.1 y 1.2.2 respectivamente. En el modelo TPG todas las tareas adyacentes tiene una relación de precedencia estricta, de tal forma que una tarea no puede empezar su ejecución hasta que sus predecesoras hayan finalizado y le hayan transmitido los datos. Esta situación se refleja con un grado de paralelismo igual a 0 en el modelo TTIG. Por otro lado, en el modelo clásico TIG no se explicita en el grafo ninguna información temporal entre tareas, y en general se asume que todas las tareas son completamente concurrentes con sus adyacentes. Esta situación queda reflejada con el grado de paralelismo igual a 1 en el modelo TTIG. Vemos pues que los valores extremos de grado de paralelismo, corresponden a situaciones extremas en la ejecución concurrente de las tareas adyacentes, que son las que se dan en los dos modelos clásicos. Además, con otros valores de grado de paralelismo se posibilita la representación de cualquier situación relativa al comportamiento temporal de las tareas adyacentes.

2.2 Proceso de generación del modelo TTIG

En los entornos distribuidos de paso de mensajes, las aplicaciones son definidas por el programador como un conjunto de procesos o tareas que ejecutan su código de forma independiente. Cada tarea constituye una unidad indivisible de cómputo que se ejecuta en un procesador concreto. Las interacciones entre las tareas tienen lugar únicamente durante la ejecución de las primitivas de paso de mensajes, en las que las tareas intercambian información mediante sentencias de envío y de recepción o bien intercambian eventos de sincronización.

Para generar el modelo TTIG de una aplicación con paso de mensajes es preciso hacer una estimación, previa a la ejecución, del valor de los parámetros asociados al grafo. Dicha estimación puede realizarse mediante los siguientes mecanismos. La forma más sencilla es que sea el mismo programador quien haga la estimación en base al conocimiento del programa que posee. Otra forma relativamente sencilla, es obtener estos valores a partir de las trazas de la ejecución del programa para una o varias muestras representativas de los valores de entrada. Finalmente, la técnica más refinada consiste en la utilización de algoritmos de aproximación estáticos que trabajan directamente sobre el código del programa [Sar89a] [ABPV91] [BDF92]. En nuestro grupo se ha utilizado la alternativa de estimación a partir de las trazas de ejecución para aplicaciones paralelas de los entornos de paso de mensajes PVM y MPI en [Cab01] y [Gon01] respectivamente, y se ha automatizado un proceso que consta de los pasos que se muestran en la Figura 2.1 [RRS+02]. En cada uno de los pasos se realizan las siguientes acciones:

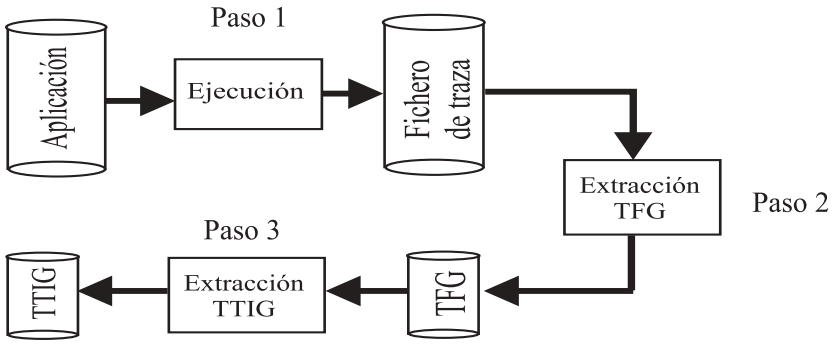

Figura 2.1: Pasos para la generación del modelo TTIG.

Paso 1. Ejecución de la aplicación para generar el fichero de traza, en el que se almacena la información relativa al cómputo de las tareas y la ejecución de las primitivas con paso de mensajes.

Paso 2. Con la información recogida en el fichero de traza se realiza un reconocimiento de la información relevante para el mapping, y se representa el comportamiento temporal resumido en el que llamamos Grafo de Flujo Temporal (TFG: Temporal Flow Graph).

Paso 3. A partir del comportamiento temporal, recogido en el grafo TFG, se extrae el modelo TTIG (Temporal Task Interaction Graph) que será el utilizado en las posteriores etapas de mapping.

El proceso automático en el que se desarrollan estos pasos está integrado en la herramienta AMEEDA, desarrollada en nuestro grupo. En el Apéndice C se resumen las principales características de dicha herramienta.

Las acciones concretas que se llevan a cabo en cada uno de estos pasos se detallan en los siguientes apartados.

2.2.1 Generación del fichero de traza

Partiendo de un programa con paso de mensajes se realiza la ejecución del mismo, para generar las trazas de ejecución que se almacenan en un fichero de traza. En dichas trazas quedan capturadas por orden de tiempo, la ejecución de las primitivas de paso de mensajes. Mediante el análisis de las mismas se pueden discernir los intervalos de tiempo en los que cada tarea realiza cómputo, y los instantes de tiempo en los que cada tarea se comunica con otras.

En nuestro grupo se han desarrollado aplicaciones para analizar la traza de los programas de los entornos de paso de mensajes PVM y MPI. En el caso de los programas codificados en C+PVM se dispone de una herramienta que interpreta las trazas de la ejecución, que han sido generadas mediante la herramienta de instrumentación de código TapePVM [Mai95] [Cab01]. Para el caso de programas en MPI se ha desarrollado un entorno que extrae el comportamiento de la aplicación partiendo de la información de las trazas que genera la herramienta Vampir [Gon01].

2.2.2 Extracción del Grafo de Flujo Temporal

A partir de la información almacenada en el fichero de traza se ordenan los eventos de cómputo y comunicación que conciernen a cada tarea por separado. De esta forma, la actividad de cada una de las tareas queda caracterizada como un conjunto de periodos de tiempo denominados *fases de cómputo*, durante los cuales la tarea ejecuta un conjunto de instrucciones de forma secuencial. Entre estas fases de cómputo existirán primitivas de paso de mensajes para intercambiar información o eventos de sincronización entre tareas.

Es importante remarcar que puesto que en esta implementación la información proviene del análisis de las trazas de ejecución, el comportamiento no determinista del programa no se captura, y todas las sentencias condicionales quedan incluidas dentro de la correspondiente fase de cómputo. En el caso más general en que la información del comportamiento se haga a partir del análisis del código, será necesario basarse en un determinado valor de probabilidad para la ejecución de las distintas alterativas que dependan de la condición a testear.

En la Figura 2.2 se muestra a modo de ejemplo el esquema de fases de cómputo y comunicaciones de un programa con tres tareas T0, T1 y T2. Las fases de cómputo están etiquetadas como A,B,..,K,L, y las comunicaciones se han representado como $send(Ti, n)$ para enviar un mensaje de volumen n a la tarea Ti, y $rcv(Ti, n)$ para recibir un mensaje con un volumen de datos igual a n de la tarea Ti.

En la Figura 2.3 se muestra de forma gráfica el patrón de cómputo y comunicaciones del programa de la Figura 2.2. En este grafo las lineas discontinuas representan fases de cómputo con su tiempo asociado y las lineas continuas representan las comunicaciones entre tareas adyacentes con el volumen de datos a transmitir. Los lazos que contienen comunicaciones se han representado con flechas discontinuas con el número de iteraciones a realizar.

Este comportamiento se codifica en el lenguaje intermedio XML. En el Apéndice A, se muestra a título de ejemplo el proceso concreto que se ha seguido para determinar el comportamiento de la tarea T2, partiendo del código C+PVM de la misma.

Partiendo del grafo que representa el comportamiento del programa se rea-

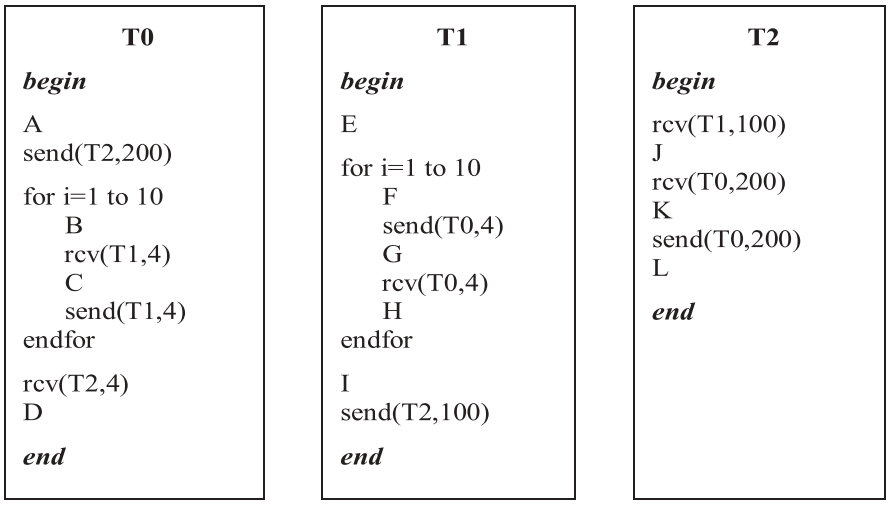

Figura 2.2: Fases de cómputo y comunicación de un programa con paso de mensajes de tres tareas.

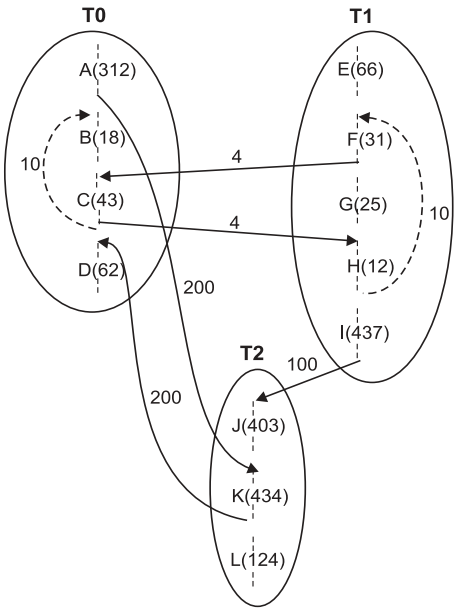

Figura 2.3: Representación gráfica del programa de la Figura 2.2.

liza un reconocimiento de la información que se considera más relevante para el proceso de mapping, a partir de la cual se construye el Grafo de Flujo Temporal (TFG: Temporal Flow Graph). Para la construcción del grafo TFG se considerarán las fases de cómputo que forman cada tarea y las primitivas de comunicación punto a punto del tipo *send* y *receive*.

En su uso más común, las operaciones de *send* son no bloqueantes, es decir, una tarea puede continuar su ejecución después de enviar un mensaje. En cambio, las operaciones de *receive* son bloqueantes, y una tarea que ejecuta una primitiva de recepción entra en espera si la tarea emisora aún no ha realizado el correspondiente envío o bien éste no ha llegado. Así pues, las primitivas de comunicación dentro de las tareas son las que marcan las relaciones de precedencia entre las mismas.

Cabe decir que las primitivas de sincronización colectivas tienen también influencia en la ejecución, pero su efecto es el de provocar una ejecución más paralela de las tareas involucradas y no se consideran para la construcción del grafo TFG.

Por lo que respecta a las estructuras iterativas, se realiza un reconocimiento de su patrón de comportamiento y se representan de forma resumida en el grafo TFG. Esto se hace así, porque para calcular el grado de paralelismo que se incluye como parámetro en el modelo TTIG, es necesario hacer una evaluación del tiempo de cómputo global en que las tareas adyacentes pueden paralelizar su ejecución. Para ello, las fases de cómputo y las comunicaciones incluidas dentro de un lazo se representan en el grafo TFG con los parámetros globales de cómputo y comunicación asociados, que se calculan mediante las siguientes pautas:

- Cada fase de cómputo dentro del lazo se representa con el tiempo de cómputo global asociado a dicha fase, considerando la ejecución de todas las iteraciones del lazo. Por ejemplo, la fase de cómputo B se representará con un tiempo de cómputo igual 180 unidades de tiempo considerando la ejecución de las 10 iteraciones del lazo.

- Cada comunicación entre tareas adyacentes incluida en un lazo, tendrá asociado un volumen de comunicación equivalente al volumen total trans-

mitido si se considera la ejecución de todas las iteraciones del lazo. Así por ejemplo, la comunicación de T1 a T0 se representará con un volumen global de 40.

La fases de cómputo y las comunicaciones incluidas dentro de un lazo, se comportan de manera repetitiva a lo largo de la ejecución del mismo. Para capturar este comportamiento repetitivo de forma global en el TFG, los lazos se representan a partir de un punto de sincronización (es decir, de una primitiva de comunicación). De esta forma se evita el posible comportamiento distinto de la primera o última iteración y queda reflejado el comportamiento estable del lazo de forma global.

La representación del lazo existente entre las tareas T0 y T1 del grafo de la Figura 2.3 se lleva a cabo a partir de la primera comunicación incluida en el mismo (el envío de T1 a T0). Entonces, el lazo entre ambas tareas se representa, en el grafo TFG, con los cómputos y comunicaciones globales tal como muestra la Figura 2.4.

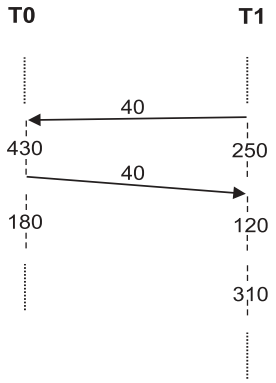

Figura 2.4: Representación para el grafo TFG, del lazo entre las tareas T0 y T1.

Es importante señalar que si bien esta es una representación simplificada del comportamiento temporal de los lazos, nos permitirá obtener el grado de paralelismo entre las tareas adyacentes en función de sus dependencias mutuas de forma correcta, tal como veremos en el Apartado 2.2.4.

A partir de los lazos representados con sus parámetros globales de cómputo y comunicación, se construye el grafo TFG que corresponde al comportamiento

temporal resumido del programa. Para el caso del programa de la Figura 2.2, su comportamiento temporal queda representado en el TFG de la Figura 2.5.

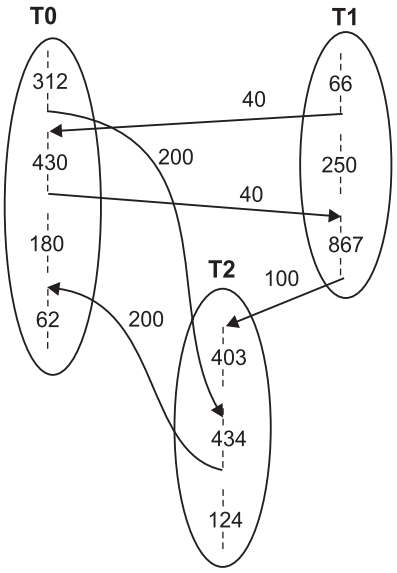

Figura 2.5: Temporal Flow Graph (TFG).

Al igual que se había dicho anteriormente, las lineas discontinuas representan fases de cómputo que deben ser ejecutadas secuencialmente con su tiempo de cómputo global. Las lineas continuas representan comunicaciones entre tareas adyacentes, con su volumen de comunicación global de datos asociado. Puede observarse en este grafo TFG, que aquellas fases de cómputo que se han de representar de forma secuencial sin interacciones entre ellas, se han puesto como una única fase con un cómputo global. Este es el caso de las fases de cómputo H, F y I de la tarea T1, que se han representado con una sola fase de cómputo con un tiempo global de 867 segundos (H+F+I=120+310+437=867).

2.2.3 Cálculo de los parámetros asociados al modelo TTIG

Obtenido el grafo TFG de un programa, se define $CP(Ti)$ como el conjunto de fases de cómputo de la tarea Ti, y $ct_m(Ti)$ será el tiempo de cómputo asociado a la fase número m de Ti. Denominamos $CM(Ti, Tj)$ al conjunto de comunicaciones representadas de Ti a Tj en el grafo TFG, y $cc_m(Ti, Tj)$ será

el volumen asociado a la comunicación número m representada de la tarea Ti a la tarea Tj.

Mediante el análisis del comportamiento temporal de las tareas, resumido en el grafo TFG, se calculan los parámetros del modelo TTIG según la siguiente metodología.

(a) Tiempo de cómputo de cada tarea.

El tiempo de cómputo, $w(Ti)$, de cada tarea Ti, se calcula como:

$$w(Ti) = \sum_{m \in CP(Ti)} ct_m(Ti) \tag{2.1}$$

$w(Ti)$ es la suma del tiempo de cómputo de todas las fases de cómputo representadas en el TFG para la tarea Ti. Para el TFG de la Figura 2.5, correspondiente al programa de la Figura 2.2, el tiempo de cómputo de la tarea T1 será:

$$w(T1) = ct_1(T1) + ct_2(T1) + ct_3(T1) = 66 + 250 + 867 = 1183$$

(b) Volumen de comunicación entre tareas adyacentes.

El volumen de comunicación, $c(Ti, Tj)$, entre dos tareas que se comunican Ti y Tj, se calcula como:

$$c(Ti, Tj) = \sum_{m \in CM(Ti,Tj)} cc_m(Ti, Tj) \tag{2.2}$$

Para dos tareas Ti y Tj, $c(Ti, Tj)$ es la suma de los volúmenes de comunicación representados en el grafo TFG de Ti a Tj. Por ejemplo, el volumen de datos a transmitir de T1 a T0 está formado por una única comunicación y será $c(T1, T0) = cc_1(T1, T0)$=40.

(c) Grado de paralelismo.

El grado de paralelismo de cada par de tareas adyacentes viene determinado por el máximo tiempo de ejecución concurrente que éstas pueden asumir. El grado de paralelismo se calcula a partir de la simulación del subgrafo aislado de ambas tareas, en el cual se consideran las dependencias entre las dos tareas. En la Figura 2.6(a) se muestra como ejemplo el subgrafo aislado de las tareas T0 y T1 del TFG de la Figura 2.5. Puesto que nuestro interés en este punto es calcular el máximo tiempo de ejecución concurrente, no se incluyen los volúmenes de comunicación en el subgrafo. Sin embargo, dichos volúmenes sí se considerarán como un parámetro adicional para ser manejado por los algoritmos de mapping a conveniencia, en función de las características de la red subyacente.

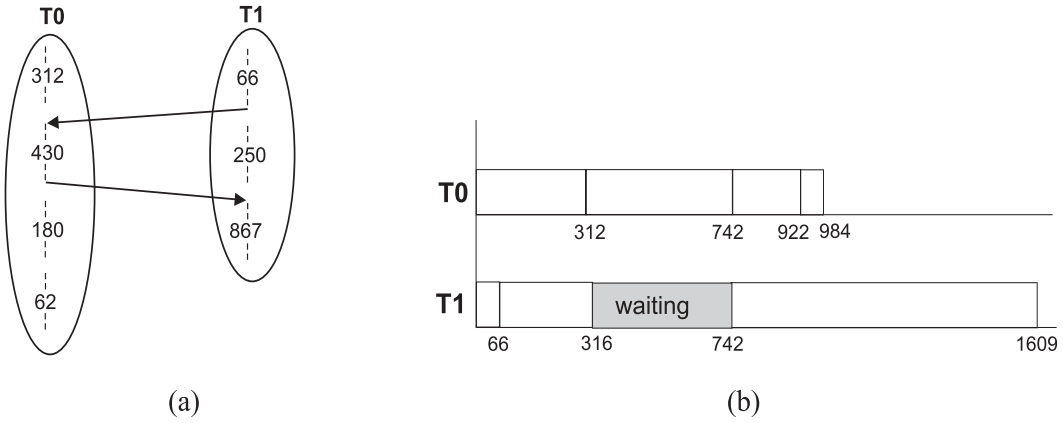

Figura 2.6: (a) Subgrafo de las tareas T0 y T1. (b) Simulación de la ejecución de este subgrafo.

En la Figura 2.6(b) se muestra la simulación del subgrafo de las tareas T0 y T1. Partiendo de los intervalos de ejecución de la simulación del subgrafo aislado, y teniendo en cuenta que el grado de paralelismo se define con respecto al tiempo de cómputo de la tarea destino en el arco, denominamos como $EI(Tj|Ti)$ el conjunto de intervalos de ejecución de Tj dependiendo de Ti, donde:

$$EI(Tj|Ti) = \{[a_{j1}, b_{j1}], [a_{j2}, b_{j2}], ..., [a_{jn}, b_{jn}]\}$$

a_{jm} y b_{jm} corresponden respectivamente al tiempo de inicio y al tiempo de finalización de la fase de cómputo número $m \in CP(Tj)$. Cabe señalar que los intervalos de $EI(Tj|Ti)$ son entre sí disjuntos y $a_{j1} \leq b_{j1} \leq a_{j2} \leq b_{j2} ... \leq a_{jn} \leq b_{jn}$. Tal como se deduce de la Figura 2.6(b), el conjunto de intervalos de ejecución de las tareas T0 y T1 serán los siguientes:

EI(T0|T1)={[0,312][312,742][742,922][922,984]}
EI(T1|T0)={[0,66][66,316][742,1609]}

El máximo tiempo de ejecución paralela de dos tareas Ti y Tj, denominado $TP(Ti, Tj)$, se define como el máximo tiempo en que las dos tareas se ejecutan concurrentemente, y se calcula a partir de sus intervalos de ejecución mediante el algoritmo de la Figura 2.7, en el que se computa el tiempo de solapamiento que se da en cada intervalo de ejecución de $EI(Ti|Tj)$ con los de $EI(Tj|Ti)$. Es evidente que $TP(Ti, Tj) = TP(Tj, Ti)$, puesto que ambos tiempos corresponden a la suma de los intervalos de ejecución común de ambas tareas. Para las tareas T0 y T1 del ejemplo desarrollado se obtiene un tiempo de ejecución paralela TP(T0,T1)=TP(T1,T0)=554 segundos, que corresponde a los tiempos de ejecución solapada de los intervalos de tiempo [0,66],[66,312],[742,922] y [922,984].

TP(Ti,Tj)=0
n_i=número de fases de cómputo de la tarea Ti
n_j=número de fases de cómputo de la tarea Tj
EI(Ti|Tj)={$[a_{i1}, b_{i1}], [a_{i2}, b_{i2}], ..., [a_{in_i}, b_{in_i}]$}
EI(Tj|Ti)={$[a_{j1}, b_{j1}], [a_{j2}, b_{j2}], ..., [a_{jn_j}, b_{jn_j}]$}
for k=1 to n_i
    for l=1 to n_j
        if max(a_{ik},a_{jl}) $\leq$ min(b_{ik},b_{jl}) then
            TP(Ti,Tj)=TP(Ti,Tj)+min((b_{ik},b_{jl})-max(a_{ik},a_{jl})
    end_for
end_for

Figura 2.7: Cálculo del máximo tiempo de ejecución paralela de dos tareas adyacentes.

Para dos tareas Ti y Tj que se comunican de Ti a Tj, el grado de paralelismo, $p(Ti, Tj)$, se calcula como:

$$p(Ti, Tj) = \frac{TP(Ti, Tj)}{w(Tj)} \tag{2.3}$$

Así pues, el grado de paralelismo da información del máximo porcentaje de su tiempo de cómputo que una tarea Tj puede realizar en paralelo respecto su adyacente Ti, teniendo en cuenta sus dependencias mutuas.

Basándonos en la definición de grado de paralelismo, puesto que las tareas T0 y T1 tienen una dependencia bidireccional, el grado de paralelismo se calculará siempre respecto al tiempo de cómputo de la tarea destino. Así, aplicando la expresión 2.3 se tiene que $p(T0, T1) = TP(T0, T1)/w(T1) = 554/1183 = 0.46$ y $p(T1, T0) = TP(T1, T0)/w(T0) = 554/984 = 0.56$.

Dicho valor representa una cota máxima en cuanto a la posibilidad de ejecución solapada de una tarea con su adyacente. Dicha capacidad de concurrencia puede que sea menor si en realidad se consideran las interdependencias existentes con el resto de las tareas del grafo. En el Apéndice B se aporta un resumen de las otras alternativas planteadas para incluir el grado de paralelismo en el modelo, y se ve como el cálculo del grado de paralelismo a partir del subgrafo aislado que considera más de dos tareas comporta una excesiva complejidad, mientras que la posibilidad de calcular el grado de paralelismo a partir de la simulación global de todas las tareas proporciona unos valores de grado de paralelismo que no siempre corresponden a la máxima capacidad de ejecución paralela.

Con los parámetros de la aplicación calculados según se ha visto previamente, se construye el grafo TTIG. Este será un grafo dirigido donde:

- Los nodos representan las tareas con su tiempo de cómputo asociado $w(T)$.
- Los arcos unen tareas que se comunican Ti y Tj, y tienen asociados dos parámetros: el volumen de comunicación $c(Ti, Tj)$, y el grado de paralelismo, $p(Ti, Tj)$, de la tarea receptora respecto de la emisora.

La Figura 2.8 corresponde al grafo TTIG que modela el comportamiento de la aplicación de la Figura 2.2. Los parámetros de este grafo se han obtenido a partir del comportamiento reflejado en el grafo TFG de la Figura 2.5, siguiendo el mecanismo desarrollado en este apartado.

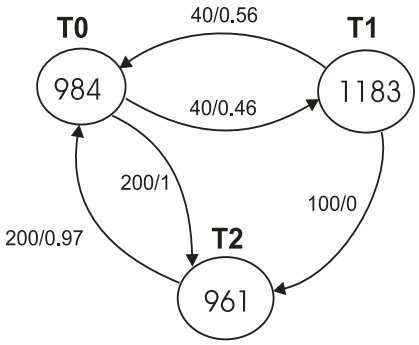

Figura 2.8: Grafo TTIG.

Como se puede observar en los valores que se incluyen en el grafo, cuando entre dos tareas existen transferencias en ambos sentidos, esto queda reflejado mediante una adyacencia bidireccional. En estos casos a cada uno de los arcos se le asocia el volumen de datos que se transmite en la dirección del arco, y el grado de paralelismo de la tarea receptora respecto de la emisora. Esto hace que cuando las dos tareas adyacentes no tienen el mismo tiempo de cómputo, el grado de paralelismo que se obtiene en una dirección sea distinto del que se obtiene en la dirección inversa, aun cuando ambos valores representan un mismo tiempo de ejecución paralela. Este es el caso, por ejemplo, de las tareas T0 y T1. Ambos grados de paralelismo implican un máximo tiempo de cómputo paralelo de 554 segundos, que representa un 56% del cómputo de T0 y un 46% del cómputo de T1.

Mediante el análisis de los distintos grados de paralelismo existentes en el grafo TTIG se observan situaciones claramente distintas respecto al comportamiento de las tareas adyacentes. Mientras la tarea T2 tiene una alta posibilidad de ejecución concurrente con T0, dado que su grado de paralelismo es 1, dicha tarea T2 tendrá que ser ejecutada de forma secuencial con T1 puesto que su grado de paralelismo es 0. Vemos por otra parte que las tareas T0 y T1 plantean una posibilidad intermedia de ejecución concurrente. Esta

información es de gran interés para una etapa posterior de mapping, puesto que aquellas tareas que no tienen posibilidad de ejecutarse de manera concurrente serán candidatas a ser asignadas al mismo procesador, mientras que las que tengan una valor alto de grado de paralelismo serán candidatas a ser asignadas a distintos procesadores para explotar su concurrencia.

2.2.4 Verificación del grado de paralelismo de los lazos

Tal como se ha visto en los anteriores apartados, los parámetros asociados al modelo TTIG se calculan a partir de la información del comportamiento del programa reflejada en el grafo TFG. Puesto que las estructuras iterativas se representan de forma resumida, en este apartado comprobamos para el ejemplo desarrollado, que el grado de paralelismo que concierne a la estructura iterativa es el mismo tanto si se calcula mediante la simulación del lazo abierto, como si se calcula mediante la simulación del lazo resumido tal y como está representado en el grafo TFG. Con esto demostramos que esta representación compactada es válida para obtener el grado de paralelismo que se ajusta a la definición del parámetro.

En la Figura 2.9 se muestra la simulación de la ejecución aislada del lazo de 10 iteraciones existente entre las tareas T0 y T1 en el grafo de la Figura 2.3. Cabe recordar que para calcular el máximo tiempo de ejecución paralela, la simulación del comportamiento se realiza teniendo en cuenta las precedencias, pero sin contemplar el coste adicional de comunicación asociado a las mismas. De esta forma, vemos en la Figura 2.9, que el comportamiento de una tarea respecto de la otra se repite de forma regular en todas las iteraciones exceptuando, si cabe, la primera y la última iteración.

Analizando el tiempo de ejecución concurrente, puede verse que en cada iteración las tareas T0 y T1 ejecutan de manera simultánea sus fases de cómputo durante 43 unidades de tiempo (que corresponde a la ejecución concurrente de las fases de cómputo con tiempo 18 y 25). Esto significa que, teniendo en cuenta su comportamiento estable, este lazo va a tener un máximo tiempo de ejecución paralela de 430 segundos si se consideran las 10 iteraciones.

Por otro lado, la simulación de la ejecución del mismo lazo, tal y como se

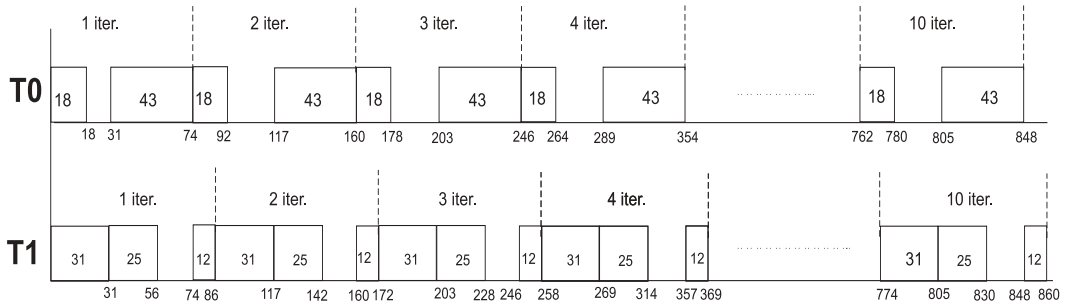

Figura 2.9: Simulación de la ejecución del lazo abierto, teniendo en cuenta sólo precedencias.

ha representado de forma global en el grafo TFG (Figura 2.4), corresponde a la Figura 2.10. Mediante el análisis de esta ejecución, puede verse que el tiempo en que las tareas T0 y T1 ejecutan de manera concurrente el lazo da un total de 430 segundos, que corresponde a los intervalos de ejecución [0,250],[430,550] y [550,610]. Vemos por lo tanto, que a partir de la simulación del lazo compactado se obtiene un máximo tiempo de ejecución paralela que es equivalente al obtenido cuando se simula la ejecución de todas las iteraciones del lazo abierto, por lo tanto el grado de paralelismo que concierne a la parte del lazo, queda calculado de forma correcta mediante la representación utilizada.

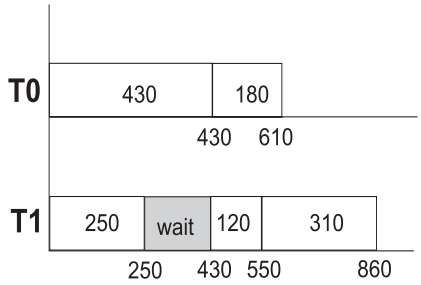

Figura 2.10: Simulación de la ejecución de la representación del lazo, teniendo en cuenta solo precedencias.

Es interesante señalar que nuestro propósito es representar el comportamiento del programa de tal manera que se obtenga un valor representativo para el grado de paralelismo, puesto que es de especial relevancia para el proceso de mapping. En este sentido, hemos probado que la forma de representar los lazos con comunicaciones es adecuada. Es evidente por otro lado, que en

una ejecución real, el comportamiento de un lazo con comunicaciones no va a ser exactamente el mismo que el de una única iteración en la que se envía todo el volumen de datos en una sola transferencia. Los costes de empaquetamiento, desempaquetamiento y transmisión de cada uno de los mensajes, van a añadir un *overhead* en las comunicaciones de cada una de las iteraciones del bucle que no se darían si la comunicación se realiza de una sola vez.

2.3 Complejidad computacional asociada al modelo TTIG

Para analizar la complejidad computacional que supone la construcción del modelo TTIG de un programa paralelo, partimos de la suposición que su comportamiento temporal ha sido representado en el correspondiente grafo TFG. De hecho, para calcular la complejidad de cualquiera de los dos modelos clásicos, TPG o TIG, sería también necesario haber obtenido previamente su grafo de comportamiento.

A partir del grafo TFG, la generación del modelo TTIG conlleva la complejidad que se deriva del cálculo de los tres parámetros que se incluyen en el grafo: tiempo de cómputo de las tareas, volumen de comunicación y grado de paralelismo entre tareas adyacentes.

Sea un grafo TFG de n tareas, donde cada tarea Ti consta de r_i fases de cómputo. Definimos $r = max\{r_1, r_2, .., r_n\}$, como el máximo número de fases de cómputo existente en cada una de las tareas del grafo. En este caso, la complejidad computacional máxima asociada al cálculo de los parámetros del grafo será la siguiente:

(a) *Tiempo de cómputo de las tareas.* Para cada tarea será necesario realizar un máximo de r sumas, por lo tanto la complejidad asociada para las n tareas del grafo será $O(rn)$.

(b) *Volumen de comunicación.* Para cada par de tareas adyacentes será necesario realizar un máximo de r sumas. En el caso de máxima conectividad cada tarea tendrá $(n-1)$ adyacentes. Entonces, el número de pares de

adyacentes será $\binom{n}{2} = \frac{n^2-n}{2}$. Por lo tanto, la complejidad máxima asociada al cálculo de los volúmenes de comunicación entre tareas será del orden de $O(rn^2)$.

(c) *Grado de paralelismo.* La simulación del grafo aislado de cada par de tareas adyacentes Ti y Tj tiene un coste de $O(r_i + r_j)$. Esto corresponde al caso en que a cada iteración del proceso de simulación, sólo una fase de cómputo es capaz de iniciar su ejecución. A partir del conjunto de intervalos de tiempo obtenidos en la simulación del subgrafo, el cálculo del grado de paralelismo necesita la ejecución de $r_i \cdot r_j$ iteraciones en el algoritmo de la Figura 2.7. Considerando la máxima conectividad entre las tareas del grafo, esto supone un coste de $O(n^2r^2)$.

De esto se deriva que el modelo TTIG puede ser calculado con un coste computacional máximo de $O(n^2r^2)$, siguiendo los pasos descritos en el Apartado 2.2.3.

Si lo comparamos con los modelos clásicos, es evidente que la generación del modelo TTIG comporta una complejidad adicional debido al cálculo del grado de paralelismo. En los modelos clásicos, los únicos parámetros que se incluyen en el grafo corresponden al tiempo de cómputo de las tareas y los volúmenes de comunicación entre tareas adyacentes. Como se ha visto, esto comporta un coste de $O(rn^2)$, frente al coste de $O(n^2r^2)$ calculado para el TTIG. Si bien éste último es más elevado, es interesante remarcar que la generación del modelo se hace de forma estática, y se realizará sólo una vez. Esta complejidad adicional se verá ampliamente compensada si el modelo TTIG permite obtener asignaciones más eficientes para aquellas aplicaciones con cómputo elevado, cuyo comportamiento se puede predecir y que se ejecutan gran número de veces.