En la Figura 6.9 se muestra el speedup obtenido en el tiempo de ejecución al ejecutar la aplicación con las distintas políticas de mapping, que pueden ser comparadas con el caso ideal. En el Apéndice D se incluyen los valores obtenidos en el tiempo de ejecución así como las asignaciones que corresponden a ocho procesadores, que son las que dan los mejores resultados.

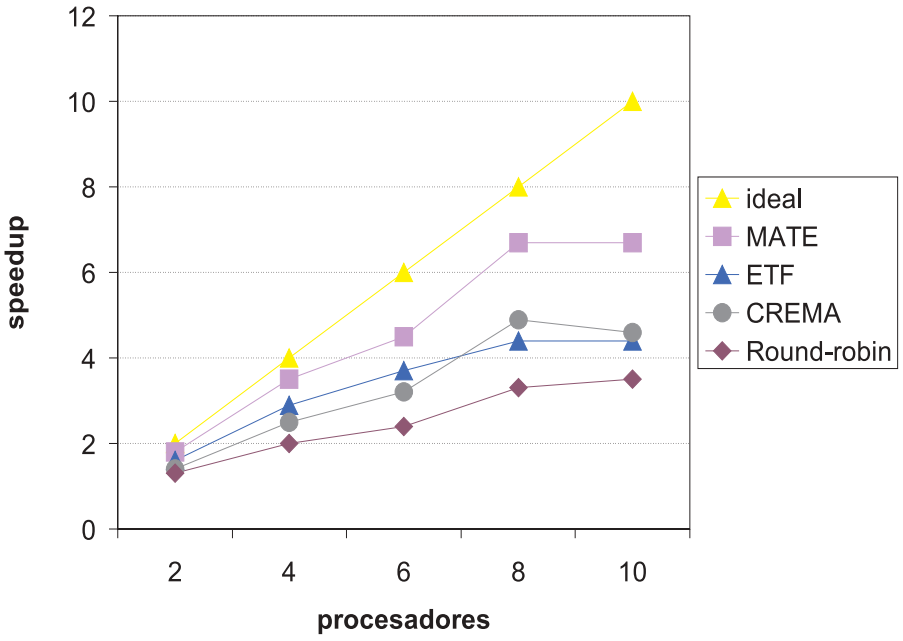

Figura 6.9: Speedup en la ejecución de la aplicación BASIZ.

Mediante el análisis de los resultados obtenidos puede verse que la política MATE basada en el modelo TTIG de la aplicación es la que proporciona mejores resultados en el *speedup*. El hecho de tener en cuenta el grado de paralelismo entre las tareas adyacentes hace que se obtengan mejores asignaciones que cuando se usa la política ETF que considera todas las tareas secuenciales. MATE también mejora los resultados de la política CREMA que considera todas las tareas completamente paralelas y sencillamente intenta balancear cómputo y minimizar comunicaciones. De las cuatro estrategias probadas, la round-robin es la que produce peores resultados debido a que no tiene en cuenta ni la estructura de tareas de la aplicación ni el comportamiento de las mismas.

Los valores obtenidos de speedup tienden a incrementarse hasta los ocho procesadores. al usar diez procesadores éste se mantiene igual o decrece. Esto es debido a la estructura de dependencias de las tareas que hace que ocho procesadores sean suficientes para explotar el paralelismo de la aplicación, que por otro lado en este caso es también el valor de cota mínima.

6.4 Estudio de la escalabilidad

Para comprobar como se comportan las estrategias de mapping definidas para el modelo TTIG a medida que se aumenta el número de procesadores utilizado en la asignación, se ha realizado un estudio representativo de la escalabilidad en el speedup, para un conjunto de grafos TTIG con distintas características.

Cada grafo TTIG se ha generado a partir del correspondiente Grafo de Flujo Temporal (TFG), que es el que se ha utilizado para realizar la simulación de la ejecución de cada una de las asignaciones.

Del mismo modo que se ha venido haciendo en los apartados previos de experimentación, los grafos utilizados pueden se pueden clasificar según sus grados de paralelismo en grafos con comportamiento arbitrario y grafos con comportamiento homogéneo, cuyas características se describen a continuación.

(a) Grafos con comportamiento arbitrario.

A este grupo pertenecen un conjunto de grafos con grados de paralelismo variando entre 0 y 1, y con topología tanto irregular como regular. Los grafos con topología regular corresponden a las topologías pipe, malla, árbol in/out y árbol binomial. En la Tabla 6.3 se muestran las características de cada uno de los grafos utilizados.

(b) Grafos con comportamiento homogéneo.

En este caso se han generado grafos con cuatro topologías regulares correspondientes al pipe, malla, árbol in-out y árbol binomial, con comportamiento homogéneo y grados de paralelismo bajo, medio y alto respectivamente. En la Tabla 6.4 se resumen las características de estos grafos.

Tabla 6.3: Características de los grafos con comportamiento arbitrario utilizados en el estudio de la escalabilidad.

Topología	Grafo	n. tareas	gr. medio	*rcc*
Irregular	G1	32	0.92	2.57
	G2	32	0.96	1.31
	G3	32	0.49	38.8
	G4	100	0.74	35.7
	G5	188	0.72	3.31
	G6	512	0.53	1.39
	G7	416	0.7	3.36
Regular	pipe	64	0.47	4,78
	malla	64	0.51	4,78
	ar_in/out	74	0.72	4.82
	arbin	64	0.43	4,68

Tabla 6.4: Características de los grafos con comportamiento homogéneo utilizados en el estudio de la escalabilidad.

		Grado de paralelismo			
Grafo	n. tareas	bajo	medio	alto	*rcc*
pipe	64	0.1	0.5	0.9	4,76
malla	64	0.1	0.5	0.9	4,76
ar_in/out	74	0.1	0.5	0.9	4.76
arbin	64	0.1	0.5	0.9	4,76

6.4.1 Resultados experimentales

Para simular la ejecución de las aplicaciones descritas en las tablas 6.3 y 6.4, se ha utilizado la herramienta de simulación ESPPADA (Entorno de Simulación de Programas Paralelos sobre Arquitecturas DistribuidAs) desarrollada en nuestro grupo [Gui01b], que realiza la simulación de programas con paso mensajes. En la Figura 6.10 se muestra el diagrama de bloques de ESPPADA.

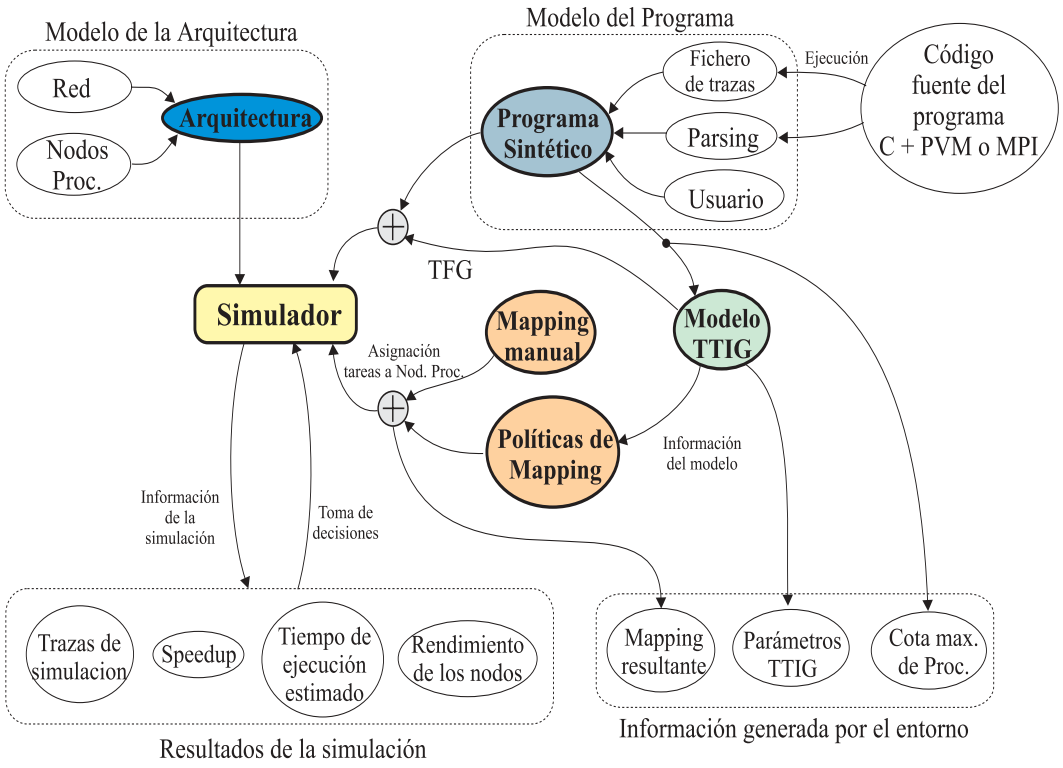

Figura 6.10: Diagrama de bloques de la herramienta ESPPADA.

Como resumen de su funcionamiento cabe señalar que acepta como entrada un programa sintético codificado en el lenguaje intermedio XML, y para la arquitectura simula el modo de funcionamiento de los sistemas tipo cluster actuales, e incorpora las técnicas no apropiativa y round-robin para el scheduling interno del procesador.

A partir del programa sintético, que puede ser obtenido con distintos mecanismos, se genera el modelo TTIG y se pueden aplicar las políticas de mapping TASC y MATE definidas para dicho modelo. Con la definición de la arqui-

tectura, el programa y la asignación, la máquina de simulación, simula la ejecución del programa, generando un conjunto de resultados e información adicional para valorar distintos aspectos de la misma. La herramienta dispone de un entorno de ventanas, de cuyas opciones más importantes se hace un resumen el Apéndice C.

La simulación de las aplicaciones descritas se ha realizado con una red tipo Ethernet y un scheduling de CPU round-robin. Se han utilizado arquitecturas con un número de procesadores entre 2 y 64 nodos, y se ha realizado el mapping de tareas usando las políticas TASC y MATE definidas par el modelo TTIG.

En el Apéndice D se muestran los valores de tiempo obtenidos al simular estas aplicaciones con distintas asignaciones y número de procesadores. En los siguientes apartados se analizan los resultados a partir del correspondiente valor de speedup.

(a) Aplicaciones con comportamiento arbitrario.

Para las aplicaciones con comportamiento arbitrario y topología irregular G1,..,G7, se ha estudiado el speedup al incrementar el número de procesadores de 2 a 64 nodos con las políticas TASC y MATE. En la Figura 6.11 se muestran los resultados obtenidos en el speedup para las dos políticas de asignación. En el nombre de cada aplicación está indicado además el número de tareas que la compone.

A la vista de los resultados se observa que, si bien los resultados muestran una ligera tendencia a ser mejores en las asignaciones de MATE, estos son muy similares para ambas políticas. Esto nos muestra que al aumentar el número de tareas y de procesadores para hacer la asignación, las diferencias entre las dos políticas se amortiguan y de hecho lo que prevalece es la habilidad del modelo en capturar el comportamiento de la aplicación.

Para las aplicaciones con topología regular y comportamiento arbitrario se ha realizado la simulación únicamente para la política MATE, puesto que se comprobó que para este tipo de topologías TASC genera unos resultados similares. En la Figura 6.12 se muestran los resultados obtenidos en el speedup para estas aplicaciones.

De los resultados obtenidos para los grafos con comportamiento arbitrario

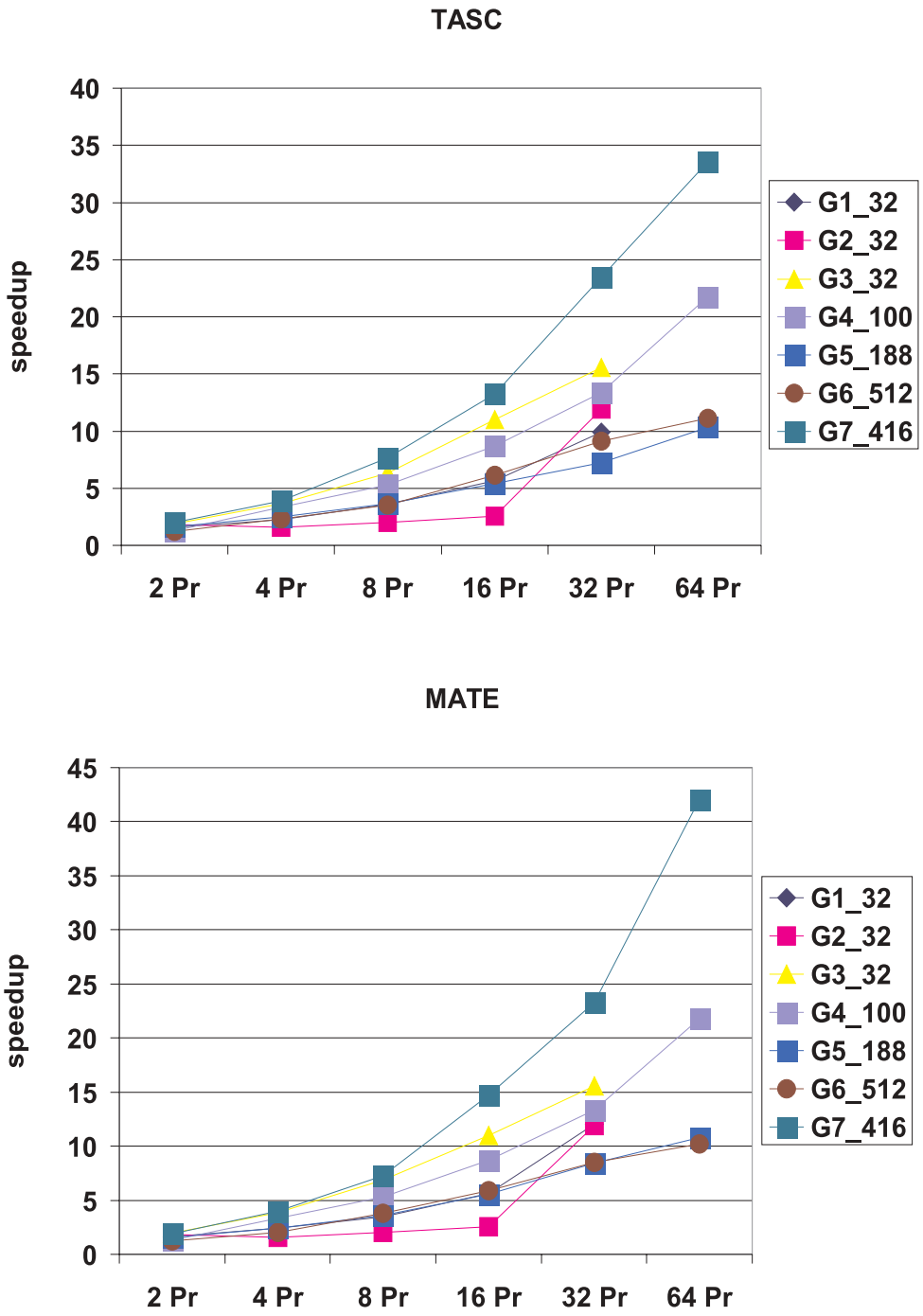

Figura 6.11: Speedup para grafos con comportamiento arbitrario y topología irregular.

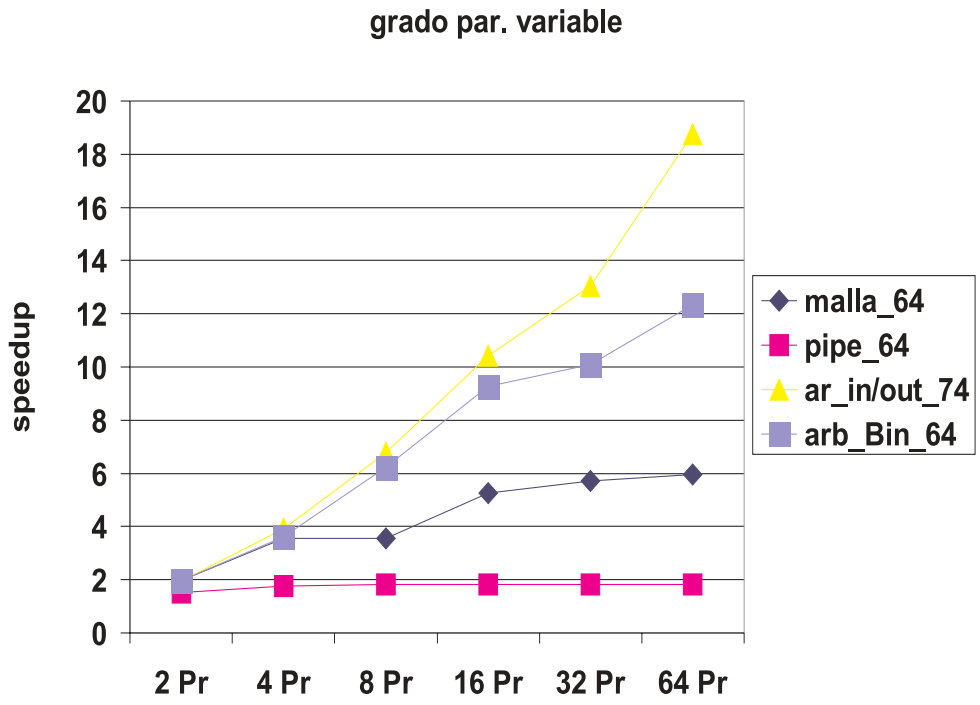

Figura 6.12: Speedup para grafos con comportamiento arbitrario y topología regular.

en las figuras 6.11 y 6.12, tanto para las topologías regulares como irregulares, se deduce que el speedup aumenta en función de la posibilidad de concurrencia de la aplicación, y que los algoritmos de mapping utilizados son capaces de detectarla. Así se ve que para el árbol in/out o el grafo G7, el speedup aumenta al ir incrementando el número de procesadores, en cambio para la topología pipe o el grafo G6 no se da prácticamente ningún incremento en el speedup.

(b) Aplicaciones con comportamiento homogéneo.

Los resultados de la escalabilidad para estas aplicaciones se muestran sólo para la política MATE, entendiendo que son también indicativos de los que se obtendrían para la política TASC.

En la Figura 6.13 se muestran los resultados obtenidos en el speedup de cada una de las aplicaciones con paralelismo bajo, medio y alto, al variar el número de procesadores de 2 a 64. Cada una de las topologías lleva indicado en el nombre el número de tareas que la compone.

A partir del análisis de los resultados vemos que las distintas topologías siguen el mismo patrón para los diversos tipos de comportamiento. Así, el árbol binomial y el árbol in-out son las que producen unas mejoras más importantes en el speedup al ir aumentando el número de procesadores, con la malla se obtiene un incremento medio y con el pipe se dan escasas o nulas ganancias al aumentar los procesadores debido a su particular estructura de interacción de tareas en cadena.

En algunos casos, y en especial para las aplicaciones con paralelismo bajo, se observa que el speedup se mantiene constante a partir de un cierto número de procesadores. Esto es debido a que la aplicación muestra poco paralelismo y no se están usando en las asignaciones todos los procesadores que posee la arquitectura.

Por lo que respecta a los distintos patrones de comportamiento, los valores del speedup van incrementando al aumentar el paralelismo intrínseco de las aplicaciones. Cuando el paralelismo es bajo se consigue un speedup máximo de 10.5 al ejecutar con 64 procesadores, cuando se trata de paralelismo medio el máximo obtenido es de 18.5, y para paralelismo alto se da un valor de speedup de 41.9 al usar los 64 procesadores en la topología árbol binomial.

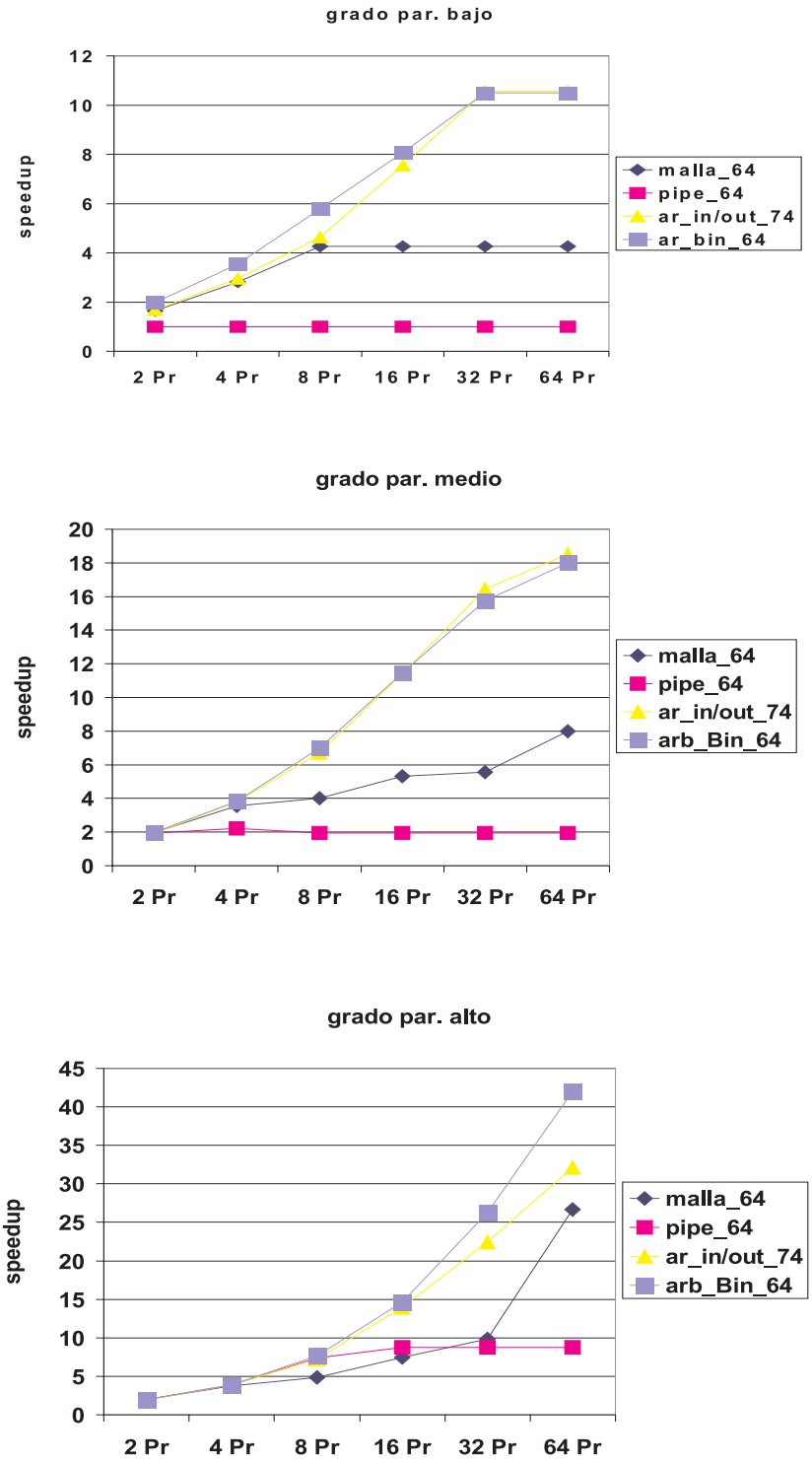

Figura 6.13: Speedup para las aplicaciones con comportamiento homogéneo.

Como resumen de la experimentación con estas aplicaciones vemos que los resultados obtenidos del mapping basado en el modelo TTIG escalan como es de esperar al ir incrementando el paralelismo. Por lo tanto, las asignaciones se llevan a cabo siempre explotando el paralelismo potencial que permite la estructura de dependencias de la aplicación, independientemente de que se vaya aumentando el número de procesadores.