Apéndice E

Aplicación del modelo TTIG en las estrategias de asignación basadas en el modelo TPG

Para aplicaciones modeladas como un Grafo de Precedencia de Tareas (TPG), existen gran cantidad de políticas de scheduling que realizan la asignación de tareas a procesadores, de las cuales se ha ralizado una breve descripción en el Capítulo 1. Teniendo en cuenta la disponibilidad que asumen del número de procesadores, los algoritmos de scheduling de TPGs pueden clasificarse en las dos categorías siguientes:

- BNP (Bounded Number of Processors). La asignación se lleva a cabo considerando una arquitectura con un número de procesadores concreto.
- UNC (Unbounded Number of Clusters). La asignación se realiza sin considerar la arquitectura sobre la que habrá que ejecutar el programa. Las decisiones que toman las heurísticas de este tipo en el proceso de scheduling, se basan en la estructura de tareas que muestra el grafo TPG. En general la estrategia que se sigue es ir formando grupos de tareas, denominados *clusters*, en función de ciertas prioridades previamente definidas. Los algoritmos UNC generan como resultado un TPG clusterizado, donde las tareas que forman parte de un mismo cluster habrá que ejecutarlas en un mismo procesador.

En los algoritmos UNC se asume la disponibilidad de un número de procesadores ilimitado, con la idea de que el uso de más procesadores servirá para mejorar el tiempo de ejecución final. De todos modos, ésta no es una suposición realista, y cuando el número de clusters generado por el UNC es mayor que el de procesadores, se hace necesaria una etapa de post-scheduling denominada *cluster-mapping*, para asignar los clusters a procesadores. La solución a la etapa de cluster-mapping es una área relativamente poco explorada en la literatura. Un aspecto interesante a estudiar, que abordamos en el presente apéndice, es la comparación de la efectividad de usar el planteamiento de los algoritmos BNP de una sola fase frente al enfoque de usar una heurísitica de dos fases formada por un algoritmo UNC más una etapa de *cluster-mapping*.

Este es, en sí mismo, un planteamiento muy amplio si se quiere abordar para la gran cantidad de algoritmos de scheduling existentes. En realidad nuestra intención en este estudio se centra en mostrar la efectividad de usar el modelo TTIG frente al TIG en la etapa de cluster-mapping, y comparar al mismo tiempo ambas soluciones con un algoritmo BNP.

E.1 Algoritmos de scheduling y cluster-mapping

Para comparar la efectividad de las distintas soluciones planteadas en el mapping de tareas de grafos TPG, dentro de los algoritmos BNP optamos por el ETF (Earliest Task First), propuesto por Hwang en [HCAL89]. El mecanismo de asignación del algoritmo ETF se ha descrito en el Apartado 6.2.

Dentro de los BNP, la elección del algoritmo ETF se debe a dos razones fundamentales. Por una parte, es un algoritmo conocido en nuestro grupo, ya que con él se han desarrollado trabajos anteriores [Roi96]. Por otra parte, produce asignaciones eficientes comparando con otros de su categoría [KA99].

Dado el grafo TPG de siete tareas {T0,..,T6} de la Figura E.1(a), y una arquitectura de dos procesadores {P0,P1} con distancia igual a 1, el algoritmo ETF comienza asignando T0 y T2 a cada uno de los procesadores. En el tiempo igual a 8, T1 está preparada puesto que su predecesora T0 ha finalizado. El mínimo tiempo de inicio de T1 es 8 en el procesador P0. Procediendo de forma sucesiva con este mecanismo, se obtiene la asignación P0:(T0,T1,T4,T6)

P1:(T2,T3,T5), cuya ejecución se muestra en la Figura E.1(b).

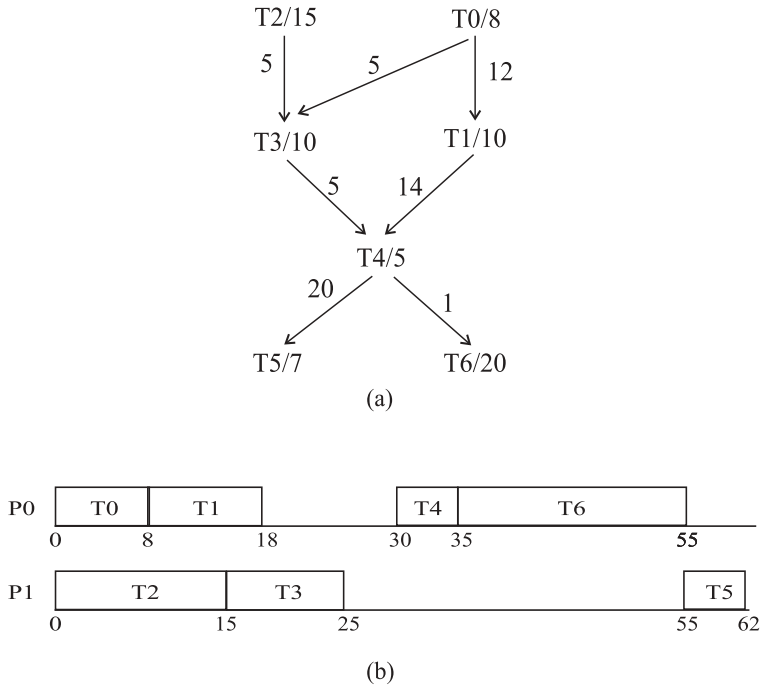

Figura E.1: (a) Grafo TPG. (b) Ejecución de la asignación obtenida usando el algoritmo ETF.

De entre los algoritmos UNC se ha escogido el DSC (Dominant Sequence Clustering) propuesto por Yang y Gerasoulis. El lector puede encontrar un estudio detallado del algoritmo en [YG94]. Aquí indicamos únicamente la idea de su funcionamiento, que se basa en ir agrupando tareas en clusters considerando la secuencia dominante (DS: Dominant Sequence). La DS es el camino crítico del TPG clusterizado. El DSC realiza una serie de pasos de agrupación intentando detectar en cada momento cuál es el conjunto de nodos y arcos que constituyen la DS del grafo y poder eliminar algún arco de ese camino agrupando sus tareas dentro de un mismo cluster, de forma que se reduzca su longitud y con ello el tiempo de ejecución final.

Aplicando el algoritmo DSC al grafo TPG de la Figura E.1(a), se obtiene el grafo clusterizado de la Figura E.2. Si se dispone de procesadores suficientes, este grafo se ejecutará asignando cada cluster a un procesador distinto. En el caso de que no se disponga de tantos procesadores como clusters habrá que

aplicar la etapa adicional de cluster-mapping a partir de TPG clusterizado.

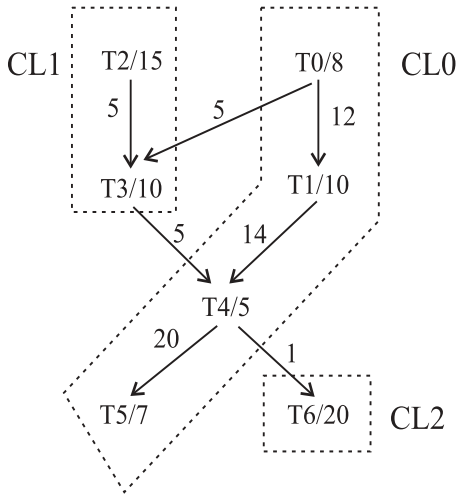

Figura E.2: Grafo TPG clusterizado.

Para resolver la etapa de *cluster-mapping*, el grafo de clusters obtenido puede ser modelado tanto como un grafo TIG como con el nuevo modelo TTIG. En la Figura E.3 se muestran los modelos TIG y TTIG que corresponden al TPG clusterizado de la Figura E.2. Como puede apreciarse, la información adicional del grado de paralelismo incluida en el TTIG, aporta una información relevante para el mapping respecto al comportamiento temporal de los disntintos clusters.

Figura E.3: (a) Modelo TIG y, (b) Modelo TTIG del grafo de clusters.

La etapa de cluster-mapping se puede resolver usando las distintas políticas utilizadas en este trabajo para cada modelo:

- Mapping TIG. Utilizando la heurística CREMA, comentada en el Apartado 6.2, la asignación del TIG de la Figura E.3(a) a dos procesadores será

P0:(CL0,CL1) y P1:(CL2). Se juntan al mismo procesador los clusters CL0 y CL1 porque así se minimizan las comunicaciones de la asignación resultante. La Figura E.4 muestra la simulación de esta asignación que da un tiempo de ejecución final de 69 unidades de tiempo.

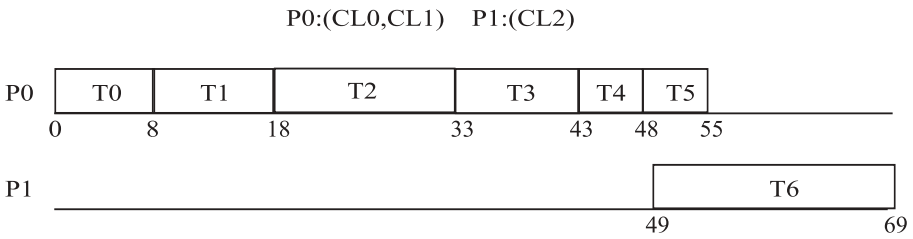

Figura E.4: Simulación de la asignación del grafo TIG.

- Mapping TTIG. Se resuelve mediante las dos políticas TASC y MATE desarrolladas en el presente trabajo. En este caso concreto la asignación de ambas políticas para el grafo TTIG de la Figura E.3(b) es P0:(CL0) P1:(CL1,CL2). Podemos ver que el cluster CL1 se asigna separado de CL0 ya que, aunque su volumen de comunicación es importante, su grado de paralelismo indica que se puede explotar la ejecución concurrente de ambas, provocando un tiempo de ejecución mejor que el anterior, igual a 56 unidades de tiempo.

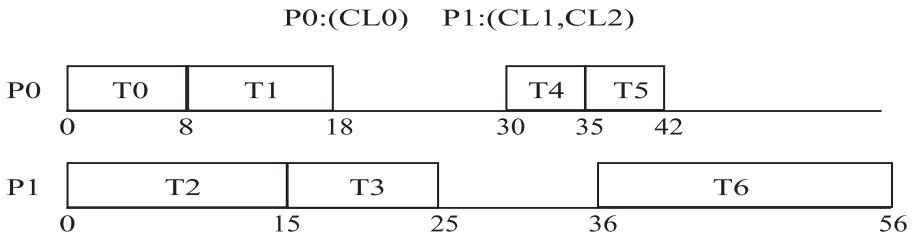

Figura E.5: Simulación de la asignación del grafo TTIG.

A través de este ejemplo hemos visto la ventaja que puede suponer usar el modelo TTIG para el grafo de clusters, y realizar la etapa de *cluster-mapping* con una estrategia basada en dicho modelo, frente a la opción de usar el modelo TIG.

E.2 Comparación de las estrategias de mapping de una y dos fases para grafos TPG

En esta sección se realiza un estudio experimental con un conjunto de grafos TPG, para valorar y comparar los resultados obtenidos en las asignaciones obtenidas mediante los siguientes tres mecanismos: (a) asignación de una sola fase, (b) asignación de dos fases modelando el grafo de clusters como un TIG y, (c) asignación de dos fases modelando el grafo de clusters como un TTIG.

Para esta experimentación se ha usado los grafos TPG que se describen a continuación

- Grafos BIN y ARB. Corresponden respectivamente a las estructuras de tareas del TPG en árbol binomial y árbol cuyas tareas se comunican con seis tareas sucesoras cada una.

- Grafos LIV1, LIV2 y LIV3. Están formados por la unión de las estructuras de los lazos de Livermore, que son representativas de estructuras de se dan en aplicaciones reales [CMS37].

- Grafos FR1,..,FR7. Corresponden a un conjunto de aplicaciones reales que fueron modeladas y utilizadas en el entormo ALPES [KCP93]. La aplicación que corresponde a cada grafo es la que se relaciona a continuación:

  FR1: Algoritmo sistólico de tiempo-espacio.

  FR2: Algoritmo sistólico de producto de matrices.

  FR3: Algoritmo de multiplicación paralela de matrices.

  FR4: Cálculo de la clausura transitiva de un conjunto de elementos.

  FR5: Algoritmo del tipo *divide and conquer*.

  FR6: Algoritmo de sustitución de Gauss.

  FR7: Algoritmo de Strassen para multiplicación rápida de matrices.

Cada uno de los grafos TPG descritos se ha clusterizado mediante el algoritmo de scheduling DSC, y se han obtenido los correspondientes modelos

TIG y TTIG a partir del grafo de clusters. En la Tabla E.1 se muestran las características resultantes de estos grafos mediante la siguiente información.

Tabla E.1: Descripción de los grafos TPG y grafos de clusters.

Grafo TPG	n. tareas	n. clusters	rcc	niveles	gr. medio
BIN	50	23	2.46	4	1
ARB	21	14	1.52	3	1
LIV1	103	65	15.5	5	0.59
LIV2	100	44	26.6	3	0.63
LIV3	109	60	19.47	5	0.57
FR1	1026	32	2.58	31	0.96
FR2	1026	32	1.31	31	0.96
FR3	1002	100	1.28	12	0.74
FR4	1002	188	1,4	15	0.72
FR5	1534	512	0.76	4	0.53
FR6	530	32	10.33	2	0.49
FR7	1483	520	1.17	5	0.44

- n. tareas. Número de tareas del grafo TPG original.
- n. clusters. Número de clusters obtenido después de aplicar el algoritmo DSC, que será a su vez el número de tareas del modelo TIG y TTIG.
- *rcc*. Relación entre cómputo y comunicación del grafo de clusters es decir, (cómputo/comunicación). Exceptuando el grafo FR5, en todos los casos se obtiene mayor tiempo de cómputo que volumen de comunicación.
- niveles. Número de niveles del modelo TTIG, entendiendo el nivel para este modelo en concreto, tal y como se ha definido en el Capítulo 4. Este valor da una indicación del orden de comienzo de las tareas y facilita la interpretación de algunos resultados como se verá más adelante.
- gr. medio. Grado de paralelismo medio de los grafos TTIG generados. Este valor da una indicación de la posibilidad de concurrencia general de las tareas de cada TTIG. Así vemos que los grafos BIN, ARB, FR1 y FR2 tienen un alto grado de paralelismo, en cambio para el resto de grafos el paralelismo medio es variado, debido a que coexisten distintos valores de grado de paralelismo en el correspondiente TTIG.

Para cada uno de los grafos se han llevado a cabo cuatro asignaciones con su posterior simulación, aplicando las siguientes políticas de una o dos fases.

1. ETF. Mapping de una fase de las tareas de grafo TPG inicial mediante el algoritmo ETF.

2. DSC+CREMA. Mapping de dos fases mediante el algoritmo DSC seguido de la política CREMA basada en el modelo TIG para el grafo de clusters.

3. DSC+MATE. Mapping de dos fases mediante el algoritmo DSC seguido de la política MATE basada en el modelo TTIG para el grafo de clusters.

4. DSC+TASC. Mapping de dos fases mediante el algoritmo DSC seguido de la política TASC basada en el modelo TTIG para el grafo de clusters.

Cada una de las políticas se ha generado para un número de procesadores variando de 2 a 32, considerando una arquitectura completamente conectada con distancia uniforme igual a 1. En el apartado final de este apéndice se muestran las tablas de los tiempos de ejecución obtenidos en cada caso, así como el tiempo serie. A partir de estos tiempos de ejecución se ha calculado el *speedup* que corresponde a cada asignación, cuyos gráficos se muestran en la Figura E.6 y la Figura E.7 respectivamente.

Mediante el análisis de los datos obtenidos, podemos ver que en general, considerando las cuatro estrategias, se obtiene un valor de *speedup* máximo que es elevado hasta los ocho procesadores, pero tiende a estabilizarse a partir de este número. Esto es debido normalmente al efecto de las dependencias entre tareas, que impiden una ejecución más concurrente y en consecuencia una mejor utilización del sistema.

Por lo que respecta al comportamiento de las distintas políticas aplicadas a cada grafo en concreto, el *speedup* sigue un comportamiento similar para los distintos valores de número de procesadores. Esto indica que la bondad de las distintas estrategias probadas depende de la estructura de dependencias del grafo inicial y el grafo de clusters generado.

El algoritmo ETF proporciona en general mejores resultados que las políticas de dos fases (exceptuando si cabe, los grafos FR2 y FR5). Esto se debe a

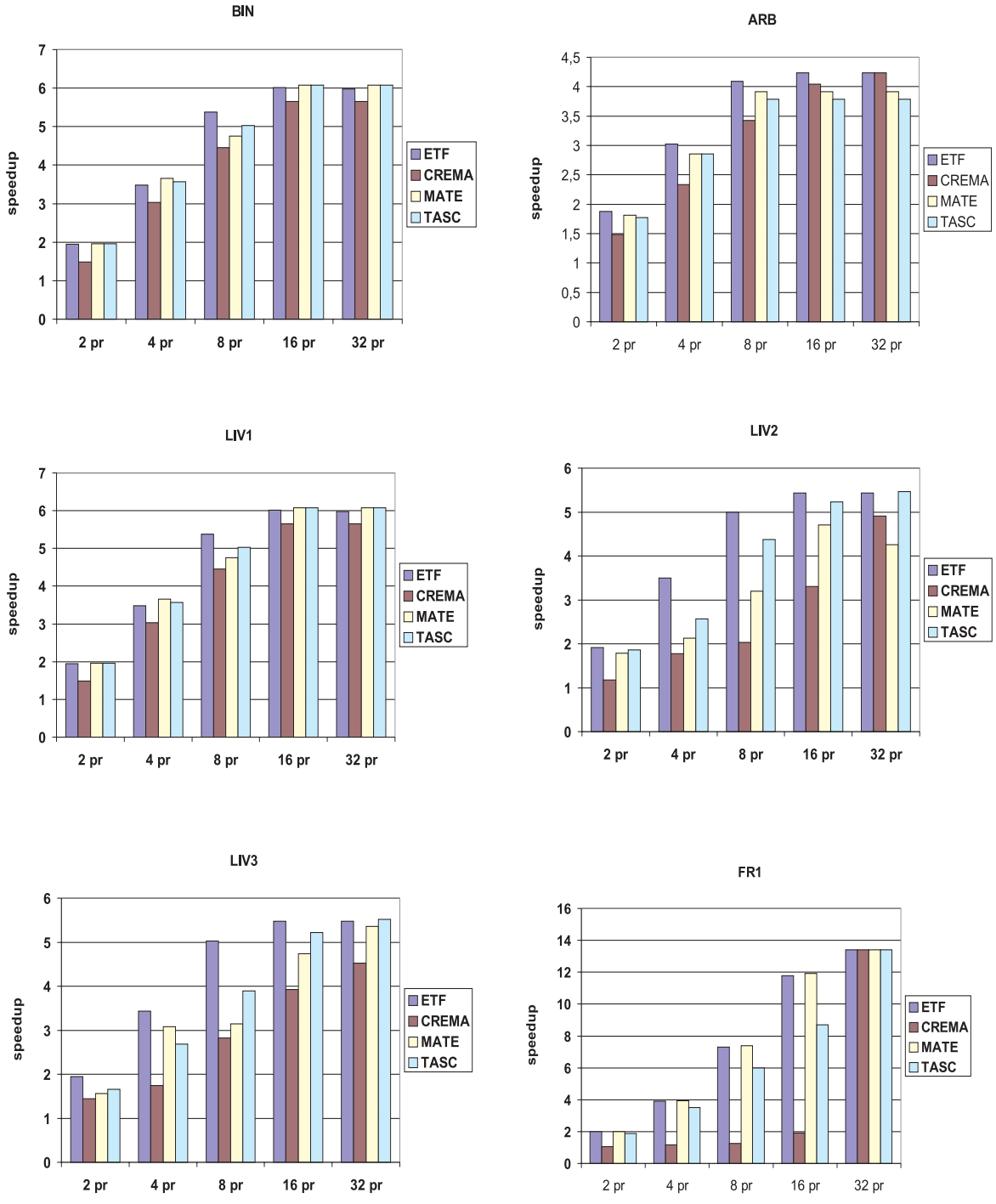

Figura E.6: Speedup para los grafos BIN, ARB, LIV1, LIV2 LIV3 y FR1.

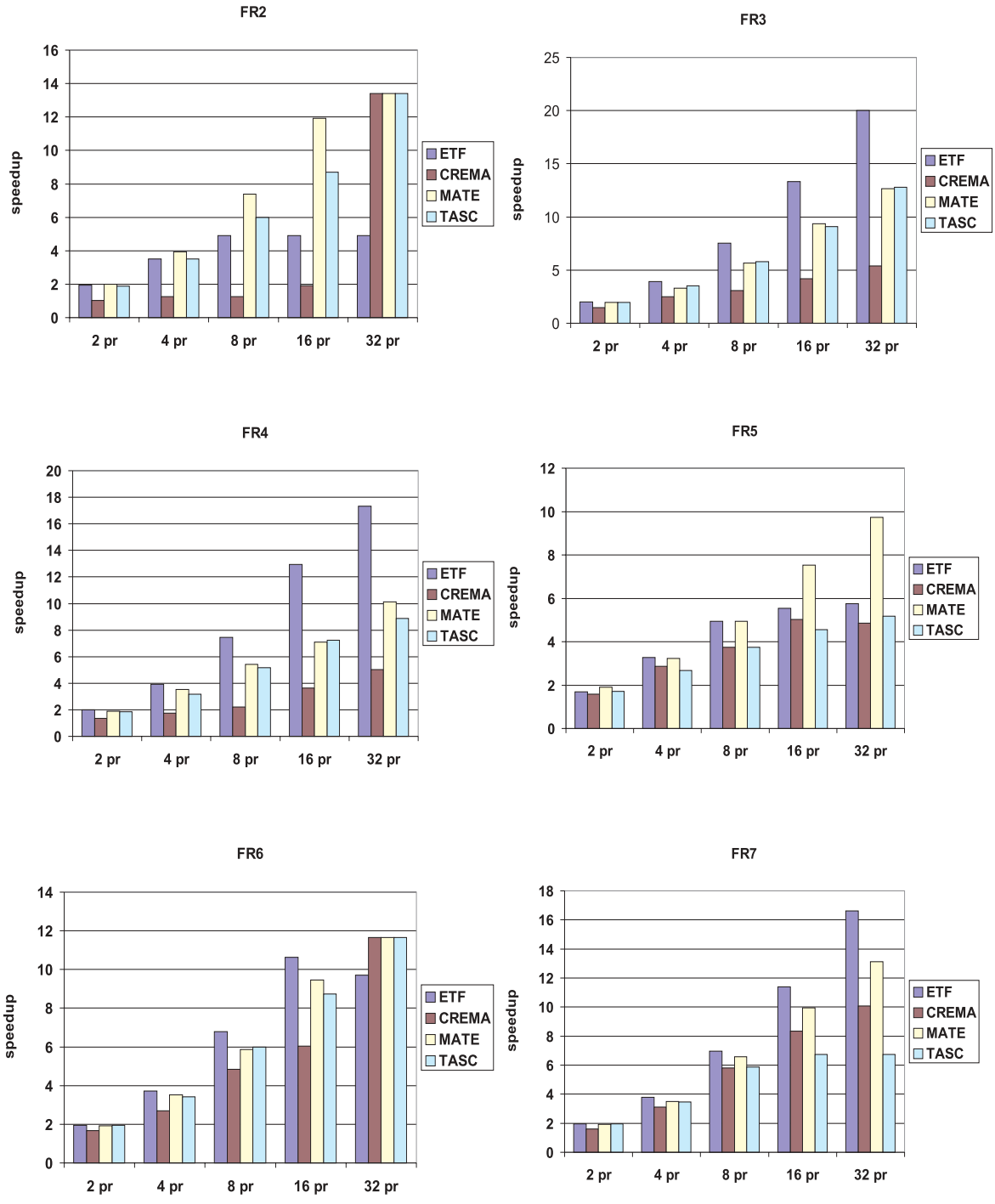

Figura E.7: Speedup para los grafos FR2,..,FR7.

que aporta mayor flexibilidad en la asignación de las tareas del TPG a nivel individual. En las estrategias de dos fases al asignar un cluster a un procesador se asignan todas las tareas del TPG incluidas en dicho cluster al mismo procesador, con lo cual se impide el inicio de algunas tareas preparadas en determinados momentos en otros procesadores.

Si se comparan las políticas de dos fases entre sí, que es el principal objetivo de este apéndice, se aprecia una clara diferencia de la estrategia DSC+CREMA basada en el modelo TIG, frente a las dos estrategias que se basan en el modelo TTIG. Con excepción de alguna asignación puntual, las estrategias basadas en el modelo TTIG generan un *speedup* claramente superior a la de CREMA. Vemos pues que el hecho de utilizar un modelo más complicado como es el TTIG, realmente ofrece ventajas en el tiempo de ejecución final frente al modelo más sencillo que es el TIG. Si se tiene en cuenta el grado de paralelismo medio del grafo de clusters, cabría esperar que en aquellos casos en que es igual o cercano a 1 (que es el máximo paralelismo) en el grafo de clusters, la política CREMA ha de generar unos resultados similares a las políticas basadas en el modelo TTIG. Esto sucede así en los grafos BIN y ARB cuyo grado de paralelismo medio es 1, en cambio el *speedup* es claramente ineficiente para los grafos FR1 y FR2 que tienen un grado de paralelismo medio igual a 0.96. La razón de este comportamiento estriba en el número de niveles del grafo. Puede verse que estos dos últimos tienen 31 niveles, lo que implica que hay una secuenciación en el orden de comienzo de las tareas, que realmente detectan los algoritmos basados en el modelo TTIG y que en cambio queda transparente en el modelo TIG.

Para las dos poíticas, MATE y TASC, basadas ambas en el modelo TTIG, puede observarse que tienen un comportamiento similar en la mayoría de los grafos estudiados. Aunque como norma general se dan mejores resultados la política MATE (hay alguna excepción como son los grafos LIV2 y LIV3), las diferencias no son muy importantes.

De los resultados obtenidos, podemos concluir por una parte que cuando se dispone de un número de procesadores pequeño en comparación al número de clusters que genera el algoritmo UNC, será más eficiente realizar la asignación con un algoritmo de una sola fase. Por otra parte, si se realiza una estrategia

de dos fases, el uso de una política basada en el modelo TTIG del grafo de clusters, en la etapa de *cluster-mapping*, aporta claras ventajas en el tiempo de ejecución final frente al uso de una política basada en el modelo TIG.

E.3 Tablas de tiempos de ejecución

En esta sección se muestran las tablas con los tiempos de ejecución obtenidos con las políticas de asignación descritas en este apéndice. Cada tabla corresponde a la asignación de uno de los grafos estudiados al ir variando el número de procesadores de 2 a 32. Se da también el tiempo de ejecución serie a partir del cual se ha calculado el *speedup*.

Tabla E.2: Grafo BIN

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	1613	2115	1608	1600
4	904	1039	860	882
8	585	706	661	626
16	523	556	517	517
32	526	556	517	517
tiempo serie: 3144				

Tabla E.3: Grafo ARB

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	192	242	199	203
4	119	154	126	126
8	88	105	92	95
16	85	89	92	95
32	85	85	92	95
tiempo serie: 360				

Tabla E.4: Grafo LIV1

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	2567	3573	2874	3291
4	1443	2586	2168	1740
8	999	2069	1245	1149
16	897	1475	1104	1143
32	897	1130	1038	893
tiempo serie: 5026				

Tabla E.5: Grafo LIV2

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	2486	4039	2667	2559
4	1366	2681	2241	1859
8	955	2352	1492	1090
16	878	1445	1013	912
32	878	972	1121	874
tiempo serie: 4772				

Tabla E.6: Grafo LIV3

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	2104	2834	2626	2468
4	1190	2340	1329	1526
8	813	1449	1302	1051
16	747	1042	864	784
32	747	903	763	741
tiempo serie: 4089				

Tabla E.7: Grafo FR1

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	5144	9638	5144	5464
4	2618	8720	2612	2932
8	1402	8108	1388	1708
16	870	5354	860	1180
32	764	764	764	764
tiempo serie: 10250				

Tabla E.8: Grafo FR2

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	5280	9944	5144	5464
2	5280	9944	5144	5464
4	2922	8094	2612	2932
8	2092	8108	1388	1708
16	2092	5354	860	1180
32	2092	764	764	764
tiempo serie: 10250				

Tabla E.9: Grafo FR3

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	5026	6860	5060	5074
4	2544	4042	3016	2856
8	1326	3246	1774	1730
16	752	2378	1068	1100
32	500	1860	792	782
tiempo serie: 10250				

Tabla E.10: Grafo FR4

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	5024	7368	5274	5390
4	2556	5786	2846	3138
8	1344	4554	1846	1940
16	774	2756	1412	1382
32	578	1990	988	1128
tiempo serie: 10010				

Tabla E.11: Grafo FR5

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	9118	9740	8024	9002
4	4682	5330	4736	5722
8	3096	4104	3098	4088
16	2772	3046	2036	3364
32	2668	3160	1574	2962
tiempo serie: 15330				

Tabla E.12: Grafo FR6

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	2730	3172	2742	2716
4	1420	1956	1500	1552
8	780	1092	902	884
16	498	876	560	606
32	545	454	454	454
tiempo serie: 5290				

Tabla E.13: Grafo FR7

n. proc.	ETF	DSC+CREMA	DSC+MATE	DSC+TASC
2	24853	30257	25467	25006
4	12912	15668	13896	14028
8	7014	8378	7431	8298
16	4281	5843	4898	7248
32	2933	4850	3721	7243
tiempo serie: 48786				

Bibliografía

[ABPV91] S. Antonelli, F. Bajardi, S. Pelagatti, and M. Vanneschi. Modeling concurrent programs for static mapping. *Proc. of Parallel Computing*, pages 357–366, Sep. 1991.

[AER93] H. Ali and H. El-Rewini. Task allocation in distributed systems: a split-graph model. *J. Combinatorial Mathematics and Combin. Computing*, 14:15–32, 1993.

[AG89] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin/Cummings, Redwood City, 1989.

[AKWS00] I. Ahmad, Y-K Kwok, M-Y Wu, and W. Shu. CASCH: A tool for computer-aided scheduling. *IEEE Concurrency*, pages 21–33, oct-dec. 2000.

[All01] E. Allué. *Desenvolupament d'una aplicació per la generació d'assignacions i l'anàlisi de simulacions en un entorn paral.lel.* Trabajo Fin de Carrera. Dep. de Informática. Univ. de Lleida., Lleida, 2001.

[BDF92] G. Balbo, S. Donatelli, and G. Franceschinis. Understanding parallel program behaviour through Petri Net models. *J. of Par. and Distr. Computing*, 15(3):171–187, 1992.

[BdKT95] P. Bouvry, J. Chassin de K., and D. Trystram. Efficient solutions for mapping parallel programs. *Proc. EuroPar Conference*, pages 379–390, 1995.

[BH98] H. E. Bal and M. Haines. Approaches for integrating task and data parallelism. *IEEE concurrency*, 6(3), July-August 1998.

[Bor01] Javier Borrás. *Desarrollo de una aplicación paralela con paralelismo funcional: detección basada en la visión humana.* Proyecto de Fin de Carrera. Dep. Informática. Universidad Autónoma de Barcelona, Bellaterra, 2001.

[BP89] J. Baxter and J.H. Patel. The LAST algorithm: a heuristic-based static task allocation algorithm. *Proc. Int'l Conf. Parallel Processing (ICPP-89)*, 2:217–222, Aug. 1989.

[Cab01] A. Cabanes. *Generació del comportament d'una aplicació a partir del processament de la seva traça d'execució.* Proyecto Final de Carrera. Dep. de Informática, Universidad Autónoma de Barcelona, 2001.

[CFS98] F. Comellas, J. Fàbrega, and A. Sánchez. *Matematica Discreta.* Edicions UPC, Barcelona, 1998.

[CKW01] J. C. Cunha, P. Kacksuc, and S. C. Winter. *Parallel program development for cluster computing (Cap. 4).* Nova Science Pub. Inc., Huntington, New York, 2001.

[CL95] M. Cosnard and M. Loi. Automatic task graph generation techniques. *Parallel Processing Letters*, 5(4):527–538, 1995.

[CMS37] L. A. Colin, T. E. Mankovic, and H. Sullivan. Performance of CHoPP. *Computer Corporation*, La Jolla, CA 92037.

[CS99] David E. Culler and Jaswinder P. Singh. *Parallel Computer Architecture. A Hardware/software Approach.* Morgan Kaufmann Pub. Inc., San Francisco, California, 1999.

[DDH+98] E. Deelman, A. Dube, A. Hoisie, Y. Luo, R. L. Oliver, D. Sundaram-Stukel, and H. Wasserman. POEMS: End-to-end performance design of large parallel adaptive computation-

al systems. *Proc. First Int. Workshop on Software Performance (WOSP)*, October 1998.

[ERS90] F. Ercal, J. Ramanujam, and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitioning. *J. Par. and Distr. Computing*, 10:35–44, 1990.

[FB73] E. B. Fernández and B. Bussell. Bounds on the number of processors and time for multiprocessor optimal schedule. *IEEE Trans. on Computers*, pages 299–305, Aug. 1973.

[Flo62] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345, 1962.

[For94] Message Passing Interface Forum. MPI: A message-passing interface standar. *Journal of Supercomputer Applications*, 8(3/4), 1994.

[Fos94] I. Foster. *Designing and Building Parallel Programs. Concepts and Tools for Parallel Software Engineering.* Addison-Wesley, 1994.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.* The MIT Press, Cambridge, Massachusets, 1994.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman and Co., S. Francisco, 1979.

[Gon01] E. González. *Análisis y procesamiento de las trazas para la obtención del modelo de comportamiento de una aplicación paralela.* Proyecto Final de Carrera. Dep. de Informática, Universidad Autónoma de Barcelona, 2001.

[Gui01a] José García Guigó. *Incorporació d'una política de mapping a l'entorn PVM, basada en un nou model de representació de programes paral·lels.* Trabajo Fin de Carrerera. Dep. Informática. Univ. de Lleida., Lleida, 2001.

[Gui01b] F. Guirado. *Entorno de simulación de programas paralelos sobre arquitecturas distribuidas.* Master tesis. Departamento de Informática. Universidad Autónoma de Barcelona, Bellaterra, 2001.

[HC97] Ch. Hui and S. T. Chanson. Allocating task interaction graphs to processors in heterogeneous networks. *IEEE Trans. on Parallel and Distr. Systems*, 8(9):908–925, 1997.

[HCAL89] J. J. Hwang, Y. C. Chow, F. D. Anger, and C. Y. Lee. Scheduling precedence task graphs in systems with interprocessor communication times. *SIAM J. Comp.*, 18(2):244–257, April 1989.

[Her91] P. Hernández. *Políticas de Scheduling Estático para Multiprocesadores.* Tesis doctoral. Departamento de Informática, Universidad Autónoma de Barcelona, 1991.

[HKK+92] S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, and C.W. Tseng. An overview of the Fortran D programming system. *Languages and Compilers for Parallel Computing. LNCS*, 589:18–34, 1992.

[HKL00] B. Hamidzadeh, L. Y. Kit, and D. J. Lilja. Dynamic task scheduling using online optimization. *IEEE Trans. on Par. and Distr. Systems*, 11(11):1151–1163, Nov. 2000.

[HQ91] P. Hatcher and M. Quinn. *Data-Parallel Programming on MIMD Computers.* MIT Press, 1991.

[Hu61] T. C. Hu. Parallel sequencing and assembly line problems. *Oper. Res.*, 9:841–848, Nov. 1961.

[JS87] J. Jenq and S. Sahni. All pairs shortest paths on a hypercube multiprocessor. *Proc. Int. Conf. on Parallel Processing*, pages 713–716, 1987.

[KA96] Y-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors. *IEEE Trans. Par. and Distr. Systems*, 7(5):506–521, May 1996.

[KA97] M. Kafil and I. Ahmad. Optimal task assignment in heterogenous computing systems. *Proc. Workshop on Heterogeous Systems in IPPS conference.*, 1997.

[KA99] Y-K. Kwok and I. Ahmad. Benchmarking and comparison of the task graph scheduling algorithms. *J. of Par. and Distr. Comp.*, 59:381–422, 1999.

[KCP93] J. P. Kitajima, C. Tron C, and B. Plateau. ALPES: a tool for the performance evaluation of parallel programs. environments and tools for parallel scientific computing. *North-Holland: Amsterdam*, pages 213–228, 1993.

[KL87] B. Kruatrachue and T. G. Lewis. Duplication scheduling heuristic: A new precedence task scheduler for parallel processor systems. *Tech Report Oregon State Univ.*, OR 97331, 1987.

[KLSZ94] C. Koelbel, D. Loveman, G. Steele, and M. Zosel. *The High Performance Fortran Handbook*. MIT Press, Cambridge M.A., 1994.

[KS84] H. Kasahara and S. Narita. Practical multiprocessor scheduling algorithms for efficient parallel processing. *IEEE Trans. on Computers*, C-33(11):1023–1029, Nov. 1984.

[KSK97] G. Karipys, K. Schloegel, and V. Kumar. PARMETIS: Parallel graph partitioning and sparce matrix ordering library. *Tech. Rep. Dep. of Computer Science. Univ. of Minessota*, 1997.

[LA87] P. J. M. Laarhoven and E. H. L. Aarts. *Simulated annealing: theory and applications*. D. Reidel Publishing Company, 1987.

[Lan98] Extensible Markup Language. XML 1.0 rec. *http://w3.org/TR1998/REC-xml-199980210*, Feb. 1998.

[LER93] T. Lewis and H. El-Rewini. Parallax: a tool for parallel program scheduling. *IEEE Parallel and Distributes Technology*, 1(2):62–72, May 1993.

[Lo88] V. M. Lo. Algorithms for static assignment and symmetric contraction in distributed systems. *Proc. IEEE Int. Conf. on Parallel Processing*, pages 239–244, 1988.

[LRG+91] V. M. Lo, S. Rajopahye, S. Gupta, D. Keldsen, M.A. Mouhamed, B. Nitzberg, J. A. Telle, and X. Zhong. OREGAMI: tools for mapping parallel computations to architectures. *Int. Journal of Parallel Programming*, 20(3):237–270, 1991.

[Mai95] E. Maillet. TAPE/PVM an efficient performance monitor for PVM applications - user guide. *ftp://tfp.imag.fr.pub/APACHE/TAPE*, 1995.

[NT93] M. G. Norman and P. Thanisch. Models of machines and computation for mapping in multicomputers. *ACM Computing Surveys*, 25(3):263–302, Sep. 1993.

[Pel98] S. Pelagatti. *Structured Development of Parallel Programs*. Taylor and Francis, London, 1998.

[Piq01] José García Piqué. *Desarrollo de una herramienta para la generación de programas sintéticos en PVM*. Trabajo Fin de Carrera. Dep. de Informática. Univ. de Lleida, Lleida, 2001.

[RCG72] C. V. Ramamoorthy, K. M. Chandy, and M. J. Gonzalez. Optimal scheduling strategies in a multiprocessor system. *IEEE Trans. Comput.*, C-21:137–146, Feb. 1972.

[Roi96] Concepció Roig. *Algoritmos de planificación para grafos de precedencia con retardos de comunicación y tiempos variables*. Master tesis, Dep. de Informática. Univ. Autónoma de Barcelona, 1996.

[RRBL01] C. Roig, A. Ripoll, J. Borrás, and E. Luque. Efficient mapping for message-passing applications using the TTIG model: a case study in image processing. *Proc. 8th Euro PVM/MPI. LNCS*, 2131:370–377, Sep. 2001.

[RRS+00a] C. Roig, A. Ripoll, M. A. Senar, F. Guirado, and E. Luque. Exploiting knowledge of temporal behaviour in parallel programs for improving distributed mapping. *6th. Int. Euro-Par Conference. LNCS*, 1900:262–271, 2000.

[RRS+00b] C. Roig, A. Ripoll, M. A. Senar, F. Guirado, and E. Luque. Modelling message-passing programs for static mapping. *IEEE Proc. 8th. Euromicro Workshop on Par. and Distr. Processing*, pages 229–236, Jan. 2000.

[RRS+01] C. Roig, A. Ripoll, M. A. Senar, F. Guirado, and E. Luque. Improving static scheduling using inter-task concurrency measures. *IEEE Proc. Int. Conf. Parallel Processing (ICPP-2001). Workshop on scheduling and resource management for cluster computing*, pages 375–381, Set. 2001.

[RRS+02] C. Roig, A. Ripoll, M. A. Senar, F. Guirado, and E. Luque. A new model for static mapping of parallel applications with task and data parallelism. *IEEE Proc. IPDPS-2002. ISBN: 0-7695-1573-8*, Apr. 2002.

[Sar89a] V. Sarkar. Determining average program execution times and their variance. *Proc. of the SIGPLAN Conf. on Programming Language Design and Implementation*, 24(7):298–312, July 1989.

[Sar89b] V. Sarkar. Partitioning and scheduling parallel programs for multiprocessors. *MIT Press, Cambridge MA*, 1989.

[SER90] P. Sadayappan, R. Ercal, and J. Ramanujam. Cluster partitioning approaches to mapping parallel programs onto a hypercube. *Parallel Computing*, 13:1–16, 1990.

[Set76] R. Sethi. Scheduling graphs on two processors. *SIAM J. Computing*, 5(1):73–82, 1976.

[SM93] S. Selvakumar and C. Silva Ram Murthy. An efficient heuristic algorithm for mapping parallel programs onto multicomputers. *Microprocessing and Microprogramming*, 36:83–92, 1992-1993.

[SRCE97] M. A. Senar, A. Ripoll, A. Cortés, and Luque E. Comparison of strategies for static mapping of parallel programs. *High Performance Computing and Networking (HPCN97) Lecture Notes in Computer Science*, 225:575–587, 1997.

[SRCL98] M. A. Senar, A. Ripoll, A. Cortés, and E. Luque. Clustering and reassignment-based mapping strategy for message-passing architectures. *IEEE Int. Par. Proc Symp&Sym. On Par. Dist. Proc. (IPPS/SPDP)*, pages 415–421, 1998.

[ST85] C. C. Shen and W. H. Tsai. A graph matching approach to optimal task assignment in distributed computing systems using minimax criterion. *IEEE Trans. Computers*, C-34(3):197–203, March 1985.

[SWP90] B. Shirazi, M. Wang, and G. Pathak. Analysis and evaluation of heuristic methods for static task scheduling. *J. of Par. and Distr. Computing*, 10:222–232, 1990.

[WG90] M. Y. Wu and D. D. Gajski. Hypertool: a programming aid for message-passing systems. *IEEE Trans. Par. and Distr. Systems*, 1(3):330–343, July 1990.

[Yan01] X. Yuan Yang. *Entorno de asignación automática de tareas a procesadores en clusters de PCs.* Proyecto de Fin de Carrera. Departamento de Informática. Universidad Autónoma de Barcelona, Bellaterra, 2001.

[YG92] T. Yang and A. Gerasoulis. PYRROS: Static task scheduling and code generation for message-passing multiprocessors. *ACM Proc. of the 1992 Int. Conf. on Supercomputing (ICS '92)*, pages 428–437, July 1992.

[YG94] T. Yang and A. Gerasoulis. DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Trans. Parallel Distr. Systems*, 5(9):951–967, Sep. 1994.

[YRR+02] X. Yuan, C. Roig, A. Ripoll, M. A. Senar, F. Guirado, and E. Luque. AMEEDA: A general-purpose mapping tool for parallel applications on dedicated clusters. *Próxima publicación en Euro-Par 2002*, 2002.