Diagramación Dinámica de Tareas de Tiempo Real Duro con Constricciones de Precedencia en Sistemas Distribuidos Multitarea-Multiprocesador

R. Cayssials, J. Orozco, J. Santos y E. Ferro.
Dep. Ingeniería Eléctrica, Instituto de Ciencias e Ingeniería de Computación,
Universidad Nacional del Sur, 8000 Bahía Blanca, Bs. As., Argentina.

E-mail: {iecayss, ieorozco, iesantos, ieferro}@criba.edu.ar
FAX: 00 54 91 553122

Resumen: En este trabajo se presenta un método para la diagramación de tareas apropiativas y cooperativas con constricciones de precedencia en un ambiente distribuido multitarea- multiprocesador. Se establecen las condiciones suficientes para la diagramación del sistema y se obtiene, como consecuencia, el algoritmo que permite analizar su factibilidad.

1. Introducción

El problema de diagramación en sistemas multitarea-monoprocesador es de complejidad polinomial y desde el trabajo liminar de Liu y Layland [1], diversos métodos han sido propuestos para resolverlo con eficacia creciente tanto desde el punto de vista de la utilización del procesador [2, 3, 4, 5] como desde el punto de vista de compartir recursos [6, 7], operando bajo distintas disciplinas de prioridades. Estas están relacionadas, en general, con los vencimientos. De ellas, la más importante, por ser un estándar *de facto* impuesto por el Departamento de Defensa de EEUU, es la denominada de Períodos Monotónicos Crecientes, PMC (*Rate Monotonic Scheduling,* RMS). En sistemas de tiempo real multitarea-multiprocesador los resultados son más escasos, quizá porque, en general, la diagramación de estos sistemas es NP-duro.

Por otro lado, tanto en sistemas mono o multiprocesador, la mayor parte de las investigaciones de la teoría clásica se encuentran en el campo de la diagramación estática donde, el algoritmo de diagramación, posee un completo conocimiento del conjunto de tareas y sus

características tales como vencimiento, tiempo de ejecución, precedencia y futuros tiempos de arribo (instante en que la tarea está lista para ser ejecutada). Sin embargo, pocos resultados se conocen en el campo de la diagramación dinámica [8]. Un algoritmo de diagramación dinámica, en este contexto, posee un conocimiento completo de las *tareas listas* para ser ejecutadas pero no conoce cuando pueden producirse nuevos arribos.

Desde el punto de vista operativo, los algoritmos de diagramación de sistemas multiprocesador pueden tener alcance global o local. En un diagramador global la disciplina de diagramación es aplicada al conjunto total de tareas sobre todos los procesadores. Si es de alcance local, existe un diagramador autónomo en cada procesador que actúa sobre las tareas que le fueron previamente asignadas.

En [1] se ha demostrado formalmente que para un sistema monoprocesador-multitarea, el peor estado de carga es el arribo simultáneo de todas las tareas. Sin embargo, en los casos de sistemas multiprocesador, cuando las tareas cooperan en su ejecución formando parte de un trabajo y son asignadas a procesadores distintos, la verificación local en cada uno de los procesadores puede encontrar el inconveniente de que los tiempos entre arribos no son exactamente conocidos.

Entre pares de tareas puede haber una relación de precedencia en el sentido de que, para comenzar su ejecución, una de ellas (*sucesora*) necesita datos producidos por la otra (*predecesora*). La llegada de dichos datos desde todas las predecesoras al procesador que aloja a la tarea sucesora define (en el contexto de este trabajo) el instante de arribo de la misma. En este caso, los tiempos entre arribos no sólo no son conocidos con exactitud sino que pueden llegar a ser menores que el período mínimo del trabajo al que pertenecen. En consecuencia, no son aplicables directamente las disciplinas de diagramación para sistemas monoprocesador.

En este trabajo, se diseña un mecanismo de diagramación local dinámico para tareas con precedencia en sistemas multiprocesador que permite utilizar resultados desarrollados para sistemas multitarea-monoprocesador. Se establecen también las condiciones de suficiencia para que un sistema sea factible mediante este mecanismo. El método de las ranuras vacías es utilizado como base de análisis de la diagramabilidad del sistema.

Debe hacerse notar que la diagramación dinámica es imprescindible para asegurar la factibilidad de un sistema multitarea-multiprocesador en el cual de ha logrado una asignación

tentativa por alguno de los métodos heurísticos o de simulación conmunmente utilizados [7, 9, 10, 11].

2. Método de las ranuras vacías.

Este método fue presentado en [12] para resolver el problema de diagramación de Redes Locales en tiempo real. Luego fue extendido, en [5] a la diagramación multitarea apropiativa en monoprocesador, aún con granularidad restringida.

El tiempo se considera ranurado y la duración de la ranura es tomada como unidad de tiempo. Las ranuras son denominadas t y numeradas 1, 2, ... Las expresiones *instante t* y *comienzo de ranura t* son equivalentes. Un conjunto de m tareas $\tau_1, \tau_2, \ldots, \tau_m$ asignadas a un procesador queda completamente especificado como $S(m)= \{(C_1, T_1, D_1), (C_2, T_2, D_2), \ldots, (C_m, T_m, D_m)\}$, donde C_i, T_i, D_i representan el máximo tiempo de ejecución, el mínimo tiempo entre arribos y el vencimiento de la tarea τ_i respectivamente.

En [1] se ha demostrado que la disciplina de Períodos Monotónicos Crecientes (PMC), es la mejor entre las de prioridades fijas en el sentido de que si un sistema es diagramable por alguna disciplina de prioridades fijas, también lo será por PMC.

En [5] se ha demostrado que la j-ésima ranura vacía que deja el sistema de m tareas luego del instante de peor estado de carga (arribo simultáneo de todas las tareas en el instante inicial), notada $e_{j(m)}$, es

$$e_{j(m)} = \text{menor}(t) \mid t = j + \sum_{h=1}^{m} C_h \left\lceil \frac{t}{T_h} \right\rceil \qquad (1)$$

En el mismo trabajo, se demuestra que, un sistema $S(m)$ es diagramable bajo PMC si y sólo si para todo $i = 1, 2, 3, .. m$, se cumple $D_i \geq e_{C_i(i-1)}$.

3. Relaciones de Precedencia

Es usual que en Sistemas de Tiempo Real (STR) resulte necesario que las tareas no sean independientes y cooperen entre sí para lograr un determinado objetivo. Cuando una tarea τ_j necesita total o parcialmente los resultados producidos por otra tarea τ_i para poder ser ejecutada, se establece una relación de precedencia, notada $\tau_i \prec \tau_j$ (τ_i precede a τ_j), la cual establece un orden

parcial sobre las tareas. Si $\tau_i \prec \tau_j$ y no existe una $\tau_l | \tau_i \prec \tau_l$ y $\tau_l \prec \tau_j$ entonces τ_i y τ_j son denominadas predecesora y sucesora respectivamente. Este modelo de computación puede ser asociado a un grafo G [11]. Cada nodo del mismo simboliza una tarea y existirá un arco dirigido desde el nodo que representa τ_i al nodo que representa τ_j si y sólo si ellas son predecesora y sucesora respectivamente. G = {Γ,P}, donde Γ es el conjunto de tareas que forman el trabajo y P es el conjunto de sus relaciones de precedencia.

Una relación de precedencia agrega complejidad a la diagramación. En [13] se ha resuelto en forma óptima, con complejidad $O(n^2)$, un problema simple de un conjunto de tareas no apropiativas, idéntico tiempo de arribo y con condiciones de precedencia. Este método, lamentablemente, no es suficiente para la mayoría de los problemas de interés en STR, en los cuales las tareas no tienen un instante conocido de arribo. En [14] se ha demostrado que el problema de diagramación de tareas con diferentes tiempos de arribo y con relación de precedencia es NP-duro. Los mejores resultados sobre precedencia fueron obtenidos a partir del trabajo sobre subclases de la relación de precedencia, alcanzándose algoritmos polinómicos para relaciones *intrees* (un solo sucesor) y *outtrees* (un solo predecesor). Estos casos son incluidos en un grupo más interesante denominado serie-paralelo que es definido en forma recursiva y sobre el cual existen soluciones eficientes, aunque, lamentablemente, los problemas no son comunes en la práctica. Si el sistema es apropiativo, se reduce la complejidad del diagramador y es solucionable en $O(n^2)$ aplicando el algoritmo de Baker [15].

4. El Modelo del sistema

El sistema está compuesto por p procesadores que deben ejecutar n trabajos. El trabajo j está integrado por $m(j)$ tareas, numeradas τ^j_1, τ^j_2,..., $\tau^j_{m(j)}$. El conjunto de n trabajos está completamente especificado por $\mathbb{J} = \{(T_1, D_1, \Lambda_1), (T_2, D_2, \Lambda_2), \ldots, (T_n, D_n, \Lambda_n)\}$ donde T_j, D_j y Λ_j representan el período mínimo, el vencimiento y el conjunto de tareas que componen el trabajo j respectivamente. El conjunto de $m(j)$ tareas que integran el trabajo j está completamente especificado por $\Lambda_j = \{(C^j_1, \Phi^j_1), (C^j_2, \Phi^j_2), \ldots, (C^j_{m(j)}, \Phi^j_{m(j)})\}$ donde C^j_i y Φ^j_i representan el máximo tiempo de ejecución de la tarea i y el conjunto de sus tareas predecesoras respectivamente.

Cada tarea está asignada a un solo procesador. Tareas del mismo trabajo pueden estar asignadas a distintos procesadores. Cada procesador tiene un conjunto de tareas asignadas. En cada procesador existe un diagramador y sólo actúa sobre las tareas asignadas a ese procesador. Esto equivale a decir que el diagramador es local en cada procesador.

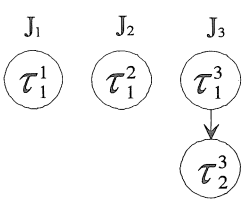
Una tarea puede activarse sólo una vez que hayan terminado de ejecutarse todas sus predecesoras. La terminación de todas las tareas predecesoras produce el arribo de la sucesora. El arribo de las tareas que no poseen predecesoras ($\Phi = \varnothing$), denominadas *raíces*, coincide con el del trabajo al que pertenecen. Las tareas que no poseen sucesora serán denominadas *hojas*.

5. El Mecanismo de Diagramación por Activación Temporizada

5.1. *Ejemplos*

Para ilustrar conceptualmente el mecanismo, se ejemplifica a continuación un mismo conjunto de trabajos con distintas condiciones de arribo y ejecución.

Sean tres trabajos, J_1, J_2 y J_3 y dos procesadores p_1 y p_2. J_1 y J_2 constan de una tarea cada uno (τ^1_1 y τ^2_1). J_3 está formado por dos tareas con relación de precedencia ($\tau^3_1 \prec \tau^3_2$). En las tablas I y II se consignan los parámetros temporales de los trabajos y de las tareas, como así también sus Λ y Φ.



Trabajo	T	D	Λ
J_1	6	6	$\{\tau^1_1\}$
J_2	8	8	$\{\tau^2_1\}$
J_3	8	8	$\{\tau^3_1, \tau^3_2\}$

Tabla I

Tarea	C	Φ
τ^1_1	3	$\varnothing$
τ^2_1	6	$\varnothing$
τ^3_1	3	$\varnothing$
τ^3_2	2	τ^3_1

Tabla II

τ^1_1 y τ^3_1 son asignadas a p_1, en ese orden de prioridad. τ^3_2 y τ^2_1 son asignadas a p_2, en ese orden de prioridad. τ^2_1 arriba en t=7, en sincronismo con τ^3_2 (Fig. 1).

Como puede verse en t=15, τ^2_1 pierde su vencimiento y el sistema no cumple con sus constricciones de tiempo. La no factibilidad proviene del arribo de τ^2_1 en un instante distinto de t=1 con lo que, el tiempo entre arribos de τ^3_2 es 5 en lugar de 8 que corresponde al período del

trabajo J_3. Este simple contraejemplo alcanza para probar que, para sistemas multiprocesador con tareas con relación de precedencia, el peor estado de carga no necesariamente es el arribo simultáneo de todos los trabajos, como sucede en los sistemas estudiados por Liu y Layland.

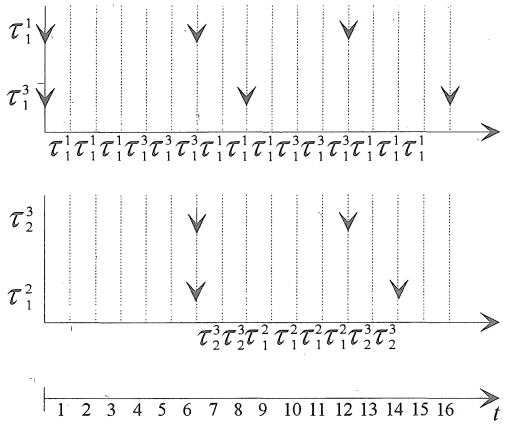


Figura 1

Analicemos ahora otro cronograma de obedecimiento (Fig. 2). En t=12, en lugar de conmutar p_2 a τ^3_2, se sigue ejecutando τ^2_1. En t=14, ésta se concluye y a *posteriori* se ejecuta τ^3_2, cumpliendo ambas con sus constricciones de tiempo.

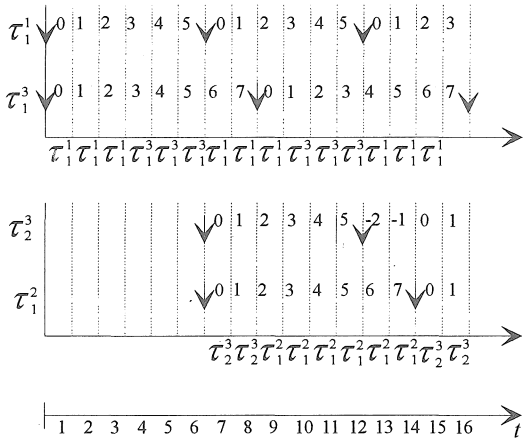


Figura 2

Nótese que, en el ejemplo, si a τ^2_1 se le asigna mayor prioridad que a τ^3_2, a pesar de tener períodos idénticos el sistema no resulta factible. Esto se debe a la relación de precedencia de τ^3_2 con τ^3_1 que adelanta el vencimiento de la primera para poder cumplir con el vencimiento de J_3.

5.2. *El Diagramador*

En lo que sigue, se denominará *Máximo Tiempo de Respuesta* de la τ^j_i, notado MTR^j_i, al máximo número de ranuras que la tarea tardará en finalizar su ejecución. Desde su arribo deberá haber ranuras suficientes para su propia ejecución y para la ejecución de tareas de mayor prioridad.

Consideremos ahora un diagramador local para cada procesador. Además de la tabla de prioridades PMC y la tabla de los MTRs, el diagramador dispone de un contador para cada tarea y opera de acuerdo a las siguientes reglas:

1. Los contadores son incrementados en una unidad en cada ranura.
2. Cada vez que una tarea predecesora termina de ejecutarse pasa a su(s) sucesora(s) un valor que es la diferencia entre el valor de su contador asociado y su MTR+1. Esta diferencia se denomina, *Valor de Inicialización del Contador de la Sucesora,* notado VICS.
3. Las tareas raíz inicializarán el contador en cero cuando se produzca su arribo.
4. La(s) tarea(s) sucesora(s) inicializan su contador con el VICS de la predecesora. Las tareas que posean más de una predecesora tomarán el menor VICS enviado por todas sus predecesoras.
5. El diagramador sólo podrá activar tareas que tengan un valor negativo en el contador si no existe ninguna con valor positivo.

El mecanismo se ilustra en la Fig. 2, donde se muestra la evolución de los contadores de las distintas tareas del ejemplo anterior. Dado que en p_1 se ejecutan τ^1_1 y τ^3_1, resulta $MTR^3_1=6$, valor que corresponde al orden de prioridades τ^1_1 / τ^3_1. En t=7, los contadores de τ^2_1 y τ^3_2 tienen el mismo valor (0). La regla de desempate asigna mayor prioridad a τ^3_2, que, en consecuencia, se ejecuta. En t=9 comienza la ejecución de τ^2_1. En t=12, el valor del contador de τ^2_1 es 6 mientras que el de τ^3_2 es -2. En consecuencia se sigue ejecutando τ^2_1 que logra completarse en t=14, antes de su vencumiento. Como puede apreciarse en el ejemplo, el mecanismo de activación temporizada por contadores produjo el comportamiento deseado.

5.3. *Formalización del método*

A continuación se demuestra formalmente el método descripto, determinando las condiciones suficientes para que el sistema resulte de diagramación factible.

Definición 1: Se denomina *Máximo Tiempo de Arribo* de τ_i^j, notado MTA_i^j, a la máxima cantidad de ranuras que la tarea podrá tener desplazado su arribo con respecto al arribo de la raíz del trabajo.

Obviamente, el MTA de toda tarea-raíz es cero.

Teorema 1

$$MTR_i^j = \text{menor}(t) \mid t = C_i^j + \sum_{\tau_h^k \in L} C_h^k \left\lceil \frac{t}{T_k} \right\rceil \tag{2}$$

donde el conjunto $\mathbb{L}$ es, a su vez, la unión de dos conjuntos. El primero compuesto por todas las tareas de mayor prioridad que la tarea *i* que no pertenecen al trabajo *j*. El segundo está compuesto por las tareas τ_k^j que perteneciendo al trabajo *j* se ejecutan en el mismo procesador, tienen mayor prioridad que la tarea *i* y $MTA_k^j \mid MTA_k^j + MTR_k^j \in (MTA_i^j, MTA_i^j + MTR_i^j]$.

Demostración:

Según los resultados de [1] el peor caso de carga en un sistema monoprocesador es el arribo simultáneo de todas las tareas de mayor prioridad. En [5] se utiliza este resultado para demostrar que si S(*i*-1) es el conjunto de tareas con mayor prioridad que la tarea *i*, la máxima cantidad de ranuras que la tarea *i* tardará en ser finalizar su ejecución es:

$$e_{C_i(i-1)} = \text{menor}(t) \mid t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil \tag{3}$$

En nuestro caso, todas las tareas que estarán en condiciones de ser ejecutadas son las de los otros trabajos (considerarlas independientes aumenta la latencia de la tarea *i*) , y las del trabajo al que pertenece la tarea *i* y que finalicen su ejecución dentro del intervalo en que se atiende a la tarea *i*. Aunque la tarea no tenga el mismo instante de arribo que la tarea *i*, el resultado no varía debido a que, por tener el mismo período, puede producir un solo requerimiento en todo intervalo si el sistema es de diagramación factible.

Por ese motivo, en la sumatoria se consideran todas las que pertenecen al conjunto $\mathbb{L}$.

Definición 2: Sea el camino λ^j_i definido por un secuencia de tareas τ^j_1, τ^j_2,..., τ^j_i, pertenecientes al trabajo i, tal que: τ^j_1 es tarea raíz y $\tau^j_h \prec \tau^j_{h+1}$ para $1 \leq h < i$.

Teorema 2

El MTA^j_i será:

$$MTA^j_i = \max \sum_{\tau_x \in \lambda^j_h} MTR^j_x \quad \forall\, \lambda^j_h \mid \tau_h \in \Phi^j_i \tag{4}$$

Demostración:

El arribo de una tarea debe esperar que todas sus predecesoras terminen su ejecución. En consecuencia, si se considera el peor tiempo para cada tarea del camino, el arribo de la tarea i tiene como cota más tardía la finalización de la ejecución del camino más lento.

Definición 3: El *tiempo máximo de ejecución del camino* λ^j_i (TEC^j_i) es definido:

$$\forall\, \lambda^j_i \qquad TEC^j_i = \max \sum_{\tau_k \in \lambda^j_i} MTR^j_k \tag{5}$$

Nótese que

$$TEC^j_i = MTA^j_i + MTR^j_i \tag{6}$$

Teorema 3

El trabajo j será factible si para toda tarea hoja i se cumple:

$$D_j \geq \max \{\, TEC^j_i \,\} \; \forall \lambda^j_i \tag{7}$$

Demostración: El trabajo se ejecutará antes de su vencimiento si el camino más lento lo hace.

Corolario

El sistema será factible con el diagramador propuesto si para todos los trabajos del sistema se cumple el Teorema 3

5.4. Algoritmo para análisis de la diagramabilidad

Las condiciones anteriores son de fácil implementación en un algoritmo.

Para todo trabajo:

I) MTA = 0 para toda tarea raíz.

II) Se calculan todos los MTR de las tareas raíces.

III) Se calcula el MTA de todas las tareas a las que se les haya calculado previamente el MTR de sus predecesoras.

IV) Se calculan los MTR de las tareas seleccionadas en III).

V) Se verifica si se incluyeron todas las tareas del trabajo en el conjunto L. Si no se incluyeron, se vuelve a IV). En caso contrario se vuelve a III) hasta concluir con todas las tareas-hoja.

VI) Se calcula el TEC (ec. 6) para todos los caminos que finalicen en una tarea-hoja del trabajo.

VII) Si el máximo TEC de los calculados en VI) es menor que el vencimiento del trabajo, éste será factible.

6. Conclusiones

El trabajo propone una solución para un problema muy poco tratado hasta la actualidad como es la operación de diagramadores dinámicos en sistemas multitarea-multiprocesador. La solución tiene como base el empleo de contadores asociados a cada una de las tareas y cuyo contenido es utilizado por el diagramador para activar o no las tareas arribadas. Se describe así mismo un algoritmo simple para determinar si dada una asignación tentativa obtenida por los métodos heurísticos o de simulación usuales, el sistema es factible.

En trabajos futuros se agregará el estudio de sistemas con variaciones en los relojes de los procesadores y retardos en las comunicaciones entre procesadores. El mecanismo permite estas modificaciones ya que los tiempos de sincronización son relativos al último arribo y no, como en otros sistemas, relativos al arribo inicial. En estos últimos, la integración del error en un período considerable hace que no se pueda acotar el error por sincronización.

Un aspecto importante es la baja complejidad del diagramador: O(n). Se aprovecha las características de prioridades fijas en el sentido de que no existen ranuras vacías si hay requerimientos pendientes.

7. Referencias

1. Liu, C.L. and Layland, J.W., "Scheduling algorithms for multiprograming in hard real time enviroments", *J. ACM*, Vol. 20, N° 1, pág. 46-61, 1973.
2. Joseph, M. and Pandya, P., "Finding Response Times in a Real-Time System", *The Computer Journal*, Vol. 29, N° 5:390-395, 1986.
3. Leung J., Whitehead J., "On the complexity of Fixed Priority Sheduling of periodic, Real -Time tasks", *Perfomance Evaluation*, Vol. 2, N° 4, pág. 237-250, 1982.
4. Lehozcky, J.P., Sha, L. and Ding, Y., "The Rate Monotonic Scheduling Algorithm: Exact characterization and average case behavior", In *Proc. Real Time Symp. IEEE CS*, Los Alamitos, CA. 1989.
5. Santos, J. and Orozco, J., "Rate Monotonic Scheduling in Hard Real-Time Systems", *Information Processing Letters*, N° 48, pág. 39-45, 1993.
6. Sha L., Rajkumar R., Lehoczky J.P., "Priority Inheritance Protocols: An aproach to Real-Time Synchronization", *IEEE Trans. Computers*, Vol. 39, N° 9,pág 1175-1185, 1990.
7. Zhao W., Ramamritham K., Stankovic J., "Preemptive Scheduling under Time and Resource Constraints", special issue on *Real-Time Systems, IEEE Trans. Computers*, Vol. 36, N° 8, pág. 949-960, 1987.
8. Stankovic J.A., Spuri M., Di Natale M., Buttazzo G.C., "Implications of Classical Scheduling Results for Real-Time Systems", *IEEE Computer* , N° 6, Vol. 28, pág 16-25, 1995.
9. Ramamritham K., Stankovic J., Shiah, "Efficient Scheduling Algorithms for Real-Time Multiprocessors Systems", *IEEE Trans. Parallel and Distr. Computting*, Vol. 4, N° 4, pág. 382-397, 1993.
10. Santos J., Ferro E., "An Heuristic Aproach to Allocating and Scheduling NP-Hard Real-Time Systems", *XX Conferencia latinoamericana de Informática,* pág. 739-746, 1994.
11. Tindell K., Burns A., Wellings A., "Allocating hard Real-Time tasks: An NP-hard problem made easy", *Real Time Systems,* pág. 145-166, 1992.
12. Santos J., Orozco J., Alimenti O., "Perfomance Evaluation of Standard Lan Protocols in Time Constrained enviroments", *IEEE INFOCOM'89*. Waterloo, Ontario, Canadá, 1989.

13.Lawler, "Optimal Sequencing of a Single Machine Subject to Precedence Constraints", *Management Science*, N° 19, 1973.

14.Lenstra J.K., Rinnooy Kan A.H.G., "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey", *Ann. Discrete Math*, N° 5, pág 287-326, 1977.

15.Ramamritham, K., "Allocation and scheduling of complex periodic task", *In Proc. 10th International Conference on Distributed Computing Systems*, pp. 108-115, 1990.