Optimización *on-line* de Sistemas de Tiempo Real con Computación Imprecisa Basados en Recompensas.

José M. Urriza, Ricardo Cayssials y Javier D. Orozco

Universidad Nacional del Sur – CONICET
Laboratorio de Sistemas Digitales
Bahía Blanca - Argentina
jurriza@criba.edu.ar

Resumen. En este trabajo se aborda el problema de optimizar, en tiempo de ejecución, la recompensa obtenida de un conjunto de tareas de tiempo real compuestas por una parte *obligatoria* de tiempo real duro y otra *opcional*. La ejecución de la parte opcional de una tarea proporciona una recompensa local según una determinada ley de recompensas no decrecientes que, en el caso tratado, es lineal. Usualmente, los sistemas de tiempo compuestos por tareas periódicas poseen una vida media muy superior a los periodos de las tareas que lo componen. Esta característica permite aplicar técnicas de optimización en tiempo de ejecución a fin de maximizar la recompensa global, garantizando la diagramabilidad del subsistema obligatorio.

1 Introducción

Los sistemas de tiempo real (STR) son aquellos en los cuales los resultados no sólo deben ser correctos desde un punto de vista aritmético-lógico, sino que además deben ser obtenidos antes de un determinado instante denominado *vencimiento*. Cuando no pueden perderse vencimientos, el sistema se denomina *duro* en oposición a los *blandos* en los cuales, son tolerales algunas pérdidas. Cuando el sistema cumple con todas las restricciones temporales, se dice que es *diagramable*. La diagramabilidad puede ser estudiada bajo diferentes aspectos: sistemas operativos, lenguajes, arquitecturas, tolerancia a las fallas y algoritmos de diagramación, son sólo algunos de ellos. [16].

En los últimos años se han encontrado importantes aplicaciones tecnológicas a los sistemas de tiempo real en los que las tareas están compuestas por una parte o subtarea dura, de ejecución *obligatoria*, y otra de ejecución *opcional*. Esta última, está diseñada de manera de producir una recompensa adicional a la ejecución de la subtarea obligatoria asociada, en términos de reducir el error del resultado obtenido por la parte obligatoria o en general, mejorar algún factor de calidad de la aplicación [3, 8, 10, 15]. Las funciones recompensa son dependientes de la aplicación y en general resultan no decrecientes siendo usualmente lineales o cóncavas (*v.g.* exponenciales o logarítmicas).

En este trabajo se presenta un método de diagramación para sistemas obligatorios-duros/opcionales-recompensados, operables en tiempo de ejecución en los cuales, la función recompensa es lineal.

Los métodos están basados en la detección de singularidades en la ejecución del sistema [12] complementados con reglas heurísticas [14].

En la Sección 2 se realiza una revisión de trabajos previos; en la Sección 3 se definen las condiciones necesarias y suficientes para la diagramabilidad de los STR; en la Sección 4 se combinan los métodos de singularidades con heurísticas apropiadas para la diagramación de sistemas obligatorio/opcional, más la descripción del método utilizado, en la Sección 5 se evalúa la performance de la optimización con estos métodos y se la compara con la producida por los mismos método sin optimizar; finalmente en la Sección 6 se presentan las conclusiones.

2 Trabajos Previos

La integración de mejor esfuerzo y diagramación por prioridades fijas fue propuesta en [1]. En [8, 10, 15] se estudian algoritmos tradicionales de refinamiento iterativo con resultados imprecisos en los que, el objetivo, es minimizar la suma ponderada de los errores.

En [4], se proponen dos formas diferentes de diagramación para una clase de problemas genéricamente denominados *IRIS* (*Increasing Reward with Increasing Service*). Un algoritmo de alta jerarquía, común a las dos formas, se ejecuta en cada activación de una tarea para determinar la cantidad de tiempo de servicio que le será otorgada; luego, un algoritmo de baja jerarquía, determina el orden en el cual las tareas son ejecutadas. Sin embargo, las tareas no se descomponen en una parte obligatoria (con un tiempo de servicio mínimo) y una opcional. Aunque pueden incluirse tareas obligatorias, no se les garantiza vencimientos duros. De hecho, se introduce como una medida de la performance, la probabilidad de perder los vencimientos antes de recibir la cantidad de servicio necesaria para satisfacer a la subtarea obligatoria.

En [3] se proponen distintos métodos que ejecutan primero las subtareas obligatorias de todo el sistema y luego seleccionan las opcionales de acuerdo a diferentes criterios (*v.g.* menor tiempo al vencimiento, menor período, etc.). El tiempo ocioso que aparece naturalmente en el sistema cuando no hay subtareas obligatorias pendientes para la ejecución se denominan ranuras de *background*. Estos métodos fueron comparados en [2] encontrando que, los mejores resultados se obtienen asignando cada ranura vacía a la subtarea opcional que mayor recompensa brinde en el instante de inspección. El método se denomina *Best Incremental Return* (BIR) y es utilizado como base de comparación en diversos trabajos.

En [2], se presenta un método *off-line* para distribuir las subtareas opcionales en las ranuras vacías del sistema. En él, se utilizan métodos de programación lineal para optimizar la función recompensa total del sistema y exige para ello que las funciones recompensa de las tareas sean continuamente diferenciables. Por otro lado, impone una restricción sobre el número de ranuras opcionales a ejecutar en cada invocación de la subtarea obligatoria, número que permanece invariante haciendo al sistema

sumamente rígido. El método no sólo introduce restricciones en este aspecto, sino que además, las partes obligatorias no pueden ser diagramadas por la disciplina de *Períodos Monotónicos Crecientes* (PMC), excepto en el caso en que los periodos sean armónicos. Esta es una limitación fuerte si se tiene en cuenta las ventajas de simplicidad y robustez que proporciona la diagramación PMC y las dificultades de diseño y tiempo ocioso que implica la necesidad de utilizar períodos armónicos.

Los métodos propuestos en este trabajo no presentan ninguno de estos inconvenientes: El subsistema obligatorio es diagramable por PMC y las funciones recompensa no necesariamente tienen que ser continuamente diferenciables. De hecho, puede utilizarse cualquier función que sea computable. Por otro lado, los métodos pueden ser usados *on-line* ya que imponen una baja carga procesamiento y un comportamiento superior al BIR. Además, en el caso de una reducción temporal o permanente en los requerimientos de ejecución de la subtarea obligatoria de una tarea, el tiempo disponible puede ser aprovechado para ejecutar la subtarea opcional de esa tarea o de cualquier otra. Adicionalmente, la función recompensa puede variar de acuerdo a condiciones internas al sistema o propias del entorno aplicativo constituyendo así, un sistema *adaptivo* [7].

3 Periodos Monotónicos Crecientes, Método de las Ranuras Vacías y Singularidades

En [9] se demostró que PMC es óptimo entre las disciplinas de prioridades fijas. Fue establecido para sus compras por el Departamento de Defensa de los EEUU y, como consecuencia, adoptado por las principales compañías del mercado: IBM, Honeywell, Boeing, General Electric, Magnavox, Mitre, General Dynamics, NASA, Paramax, McDonnell Douglas, etc. [11]. Es por esto una norma de facto, al menos en los Estados Unidos, y su utilización resulta atractiva, especialmente en sistemas aplicados en busca de mercado.

Varios métodos han sido propuestos para determinar las condiciones de diagramabilidad de los STR operando bajo PMC [6, 9, 13]. Todos tienen en común, el hecho de que la relación de orden de prioridades está basada en periodos crecientes, con alguna regla adicional para dirimir los empates. En el Método de las Ranuras Vacías [13], el tiempo se considera ranurado y la duración de una ranura es tomada como la unidad de tiempo. Las ranuras se notan t y se numeran 1, 2, ... La expresión *al comienzo de la ranura t* e *instante t* son equivalentes. El sistemas es apropiativo y las tareas pueden ser desalojadas del procesador únicamente al comienzo de la ranura.

Un conjunto $\mathbf{S}(n)$ de n tareas independientes periódicas apropiables se encuentra completamente especificado por $\mathbf{S}(n)=\{(C_1, T_1, D_1), (C_2, T_2, D_2),..., (C_n, T_n, D_n)\}$, donde C_i, T_i y D_i, indican el máximo tiempo de ejecución, el período o, si es variable, el mínimo tiempo de interarribo y el vencimiento de la tarea i respectivamente. En los sistemas del mundo real, en general $D_i \leq T_i$. En [13] se demostró formalmente que $\mathbf{S}(n)$ es diagramable por PMC sss:

$$\forall i \in (1,2,....,n) \qquad T_i \geq D_i \geq \text{menor } t \mid t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil$$

En otras palabras, si antes de la siguiente activación, cada tarea encuentra suficientes ranuras libres para ejecutarse y dar lugar a las tareas de mayor prioridad. Las ranuras quedan vacías únicamente si no hay tareas con ejecución pendiente.

En [12] se propusieron dos métodos basados en singularidades, instantes especiales en la evolución del sistema, para optimizar la diagramabilidad de sistemas mixtos en los cuales además de tareas duras hay tareas blandas y otras que no tienen vencimientos (no duras).

Una singularidad s, se define como una ranura en la cual todas las tareas pertenecientes al sistema $\mathbf{S}(n)$ que fueron activadas en el intervalo [1, (s-1)], han sido ejecutadas. Nótese que s-1 puede ser tanto una ranura vacía como una ranura en la cual se terminó de ejecutar la última de las tareas periódicas pendientes. s es una singularidad aún en el caso de que en t=s, se generen nuevas instancias de las tareas periódicas. Una singularidad s_i es una ranura en la cual todas las tareas en el subsistema $\mathbf{S}(i)$, con activación en el intervalo [1, (s-1)], han sido ejecutadas. El método basado en la detección de s se denomina *Detección de Singularidad Simple* (DSS), mientras que el basado en la detección de todos las s_i se denomina *Detección de Singularidad Múltiple* (DSM). Los dos métodos propuestos solo utilizan *Detección de Singularidad Múltiple* dado que se tiene un mejor control de la posible ubicación de las ranuras disponibles.

4 Sistemas Obligatorios k-PMC / Opcionales Recompensados

En lo que sigue, una tarea genérica, notada τ_i , $i \in \{1, 2, ..., n\}$, puede ser vista como $\tau_i = M_i \cup O_i$, donde M_i y O_i indican las subtareas o partes obligatorias y opcionales, con tiempos de ejecución m_i y o_i respectivamente. Obviamente la condición de diagramabilidad anterior debe ser aplicada sólo a la parte obligatoria del sistema que será atendida por PMC.

El número máximo de ranuras disponibles para la ejecución de opcionales es independiente de la disciplina de diagramación que se utilice: siempre será el número de ranuras no utilizadas por el subsistema obligatorio. M denota el mínimo común múltiplo de los periodos de las n tareas y se denomina hiperperíodo del sistema. La función *trabajo* $W_n(M)$ se define como

$$W_n(M) = \sum_{i=1}^{n} m_i \frac{M}{T_i}$$

y representa, el número de ranuras necesarias para procesar el subsistema obligatorio en el intervalo [1, M]. Es evidente que el número de ranuras vacías disponibles para la atención de las partes opcionales está dado por $M - W_n(M)$, y debido a que las tareas son periódicas, es suficiente con estudiar al sistema en el hiperperíodo.

Si $M - W_n(M) < \sum O_i$, el sistema es sobresaturado y únicamente una porción de las subtareas opcionales podrá ser ejecutada. De hecho, el valor máximo de recompensa que puede ser alcanzado en un sistema durante un hiperperíodo, es función del número de ranuras vacías disponibles para la ejecución de opcionales. La manera más sencilla de aprovechar este tiempo ocioso, usualmente denominado *slack-time,* es en forma pasiva, es decir, cuando está naturalmente disponible por ausencia de

requerimientos de ejecución (*background*). Por el contrario, los algoritmos presentados en este trabajo para atacar el problema de la asignación de ranuras vacías a subtareas opcionales, son métodos activos que redistribuyen la disponibilidad del tiempo ocioso a lo largo del hiperperíodo a fin de lograr la mejor distribución del tiempo ocioso de acuerdo a cierta función objetivo.

Sea $\mathbf{S}(n)=\{(m_1, o_1, T_1, D_1), (m_2, o_2, T_2, D_2),\ldots, (m_n, o_n, T_n, D_n)\}$ el conjunto de n tareas independientes periódicas apropiables a ser diagramado con partes obligatorias PMC y opcionales. Éstas últimas deben ser ejecutadas antes de que se vuelva a generar un nuevo requerimiento de sus partes obligatorias, tratando de maximizar la recompensa total en el hiperperíodo. Las funciones recompensa son no decrecientes, lineales o cóncavas, y están afectadas por una función de depreciación.

Un subsistema obligatorio es k-PMC diagramable si:

$$\forall i \in (1,2,....,n) \qquad T_i \geq D_i \geq \text{menor } t \mid t = m_i + k + \sum_{h=1}^{i-1} m_h \left\lceil \frac{t}{T_h} \right\rceil \quad (1)$$

En general, k será una cota inferior común de $\{k_i\}$. k_i se define como:

$$k_i \geq k \mid T_i \geq D_i \geq \text{menor } t \mid t = C_i + k + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil \quad (2)$$

En [12] se demostró que si el subsistema obligatorio es k-PMC diagramable, pueden asignarse las k ranuras del intervalo $[s, (s+k-1)]$, para ejecutar tareas del subsistema opcional. El método denominado DSM, refina al anterior al utilizar una cota k_i, para cada subtarea m_i en lugar de una cota global.

En [14] las nociones de diagramabilidad k-PMC y singularidades, combinadas con heurísticas, fueron utilizadas para el desarrollo de cuatro métodos de diagramación *on-line* de sistemas obligatorios k-PMC/opcionales-recompensadas. En este trabajo sólo utilizaremos dos de estos métodos, llamados DSM1 y DSM2. La heurística se incorpora porque el simple avance del *slack-time* no es suficiente para mejorar la performance. Habrá casos en los cuales subtareas opcionales de alto valor de recompensa estarán asociadas a subtareas obligatorias de baja prioridad y viceversa. En esos casos, avanzar en la ejecución de las subtareas opcionales puede producir resultados adversos si la conclusión de una obligatoria de alta prioridad, habilita la ejecución de una opcional de baja recompensa, bloqueando la ejecución posterior de una de opcional de mayor recompensa, asociada a una obligatoria de menor prioridad.

En la primera de las heurísticas propuestas, se evita la ejecución de una opcional de baja recompensa cuando la única razón para hacerlo es que la subtarea obligatoria de alta prioridad asociada a la misma haya sido ejecutada. Si una subtarea opcional de alta recompensa no puede ser ejecutada, porque su parte obligatoria aún no lo ha sido, se ejecutan las subtareas obligatorias siguiendo la disciplina PMC. En algún momento se concluirá la ejecución de la subtarea obligatoria que habilite la ejecución de la subtarea opcional de alta recompensa utilizando ranuras del *slack-time* antes del instante en que lo harían de ejecutarse sólo en *background*.

La segunda heurística en cambio, produce un adelantamiento del *slack-time* para ejecutar tanto subtareas obligatorias (aun quebrando el ordenamiento por PMC) como

opcionales. Esto pretende ejecutar lo más pronto posible las partes obligatorias asociadas a opcionales de recompensa alta.

En lo que sigue se describen los dos algoritmos paso por paso. DSM1 y DSM2 se implementan por medio de *n* contadores, AC_i, $i \in \{1,2,...,n\}$; el contenido de los mismos se nota AC_i.

DSM1:

1) $\forall g \in \{1,2,...,i\}$, $AC_g = k_g$ en $t = s_i$.

2) ***If*** $\forall i \in \{1,2,...,n\}$, $AC_i \neq 0$ y no hay pendiente de ejecución una obligatoria con una parte opcional que posea mayor recompensa que la subtarea opcional a ser ejecutada, ***then*** la subtarea opcional se ejecuta, ***else*** se sigue el ordenamiento PMC.

3) $\forall i \in \{1,2,...,n\}$, AC_i se decrementa en una unidad por cada ranura asignada a una parte opcional.

DSM2:

1) $\forall g \in \{1,2,...,i\}$, $AC_g = k_g$ en $t = s_i$.

2) ***If*** $\forall i \in \{1,2,...,n\}$,$AC_i \neq 0$ y no hay obligatorias pendientes de ser ejecutadas con partes opcionales asociadas con recompensa mayor a la de la subtarea opcional a ser ejecutada, ***then*** la subtarea opcional es ejecutada, ***else*** la subtarea obligatoria con opcional asociada de mayor recompensa se ejecuta aún cuando con ello se viole el ordenamiento PMC.

3) ***If*** se ejecuta una opcional, ***then*** $\forall i \in \{1,2,...,n\}$, AC_i se decrementa en una unidad, ***else*** únicamente se decrementan en una unidad los contadores de las tareas invertidas al violar el ordenamiento PMC.

Debe notarse que, dado que las ranuras vacías constituyen una singularidad, los contadores serán recargados también en ellas.

Los métodos preservan el ancho de banda pues las ranuras libres que se pueden ejecutar luego de cada singularidad no se pierden en caso de no ser utilizadas ya que pueden ser usadas más tarde. Una opcional puede esperar hasta la próxima activación de la subtarea obligatoria o hasta la próxima singularidad. En ese instante sin embargo los contadores son recargados, contando nuevamente con *k* ranuras disponibles. La actualización de los contadores es la única sobrecarga que imponen los métodos DSM.

4.1 Método Metaheurístico para Sintonizar k_i

La optimización propuesta consiste en modificar el valor de recarga de estos contadores entre los valores [1, k_i] modificando la distribución del *slack* en el hiperperíodo. Dado que la operación de los contadores de cada tarea no resulta independiente entre sí, un simple experimento permite mostrar que sólo algunos contadores dominan la dinámica del sistema y que, aún siendo estos los de máximo valor, no necesariamente mejoran la recompensa final. Este trabajo busca, por medio de un algoritmo metaheurístico como el recocido simulado (RS), mejorar la recompensa global del sistema, ajustando los valores de k_i en lugar de utilizar la cota

superior que produce la expresión (2). Debido a que el sistema posee tantos parámetros a ajustar como tareas, se realizó una adaptación del método de RS introduciendo en cada iteración una selección aleatoria del parámetro k_i a variar.

En un principio, se corre el sistema con los valores de k_i que produce la expresión (2) para calcular la recompensa total que proporciona la solución inicial. Inicializado el método RS se elige de manera aleatoria una tarea dentro del intervalo [1, n] a quien se asigna un valor aleatorio a su k en el intervalo [0, k_i]. Además se decide de manera aleatoria si este valor será incrementado o decrementado en la próxima iteración. Con ese valor, se corre una simulación durante un hiperperíodo, al final del cual, se compara con el valor de la recompensa obtenida previamente. Si la nueva recompensa es superior, se adopta el nuevo valor de k_i. En caso contrario, de la aplicación de la expresión: $x < e^{-\frac{f(k_i)-f(k_0)}{t}}$ con x un valor aleatorio entre [0,1], se decide si se simula otro hiperperíodo incrementado o decrementado k_i. En el caso contrario se reinicia el proceso y se selecciona aleatoriamente otra tarea dentro del intervalo [0, n]. Si alguno de los extremos del intervalo [0, k_i] es alcanzado, se procede como el caso anterior.

5 Evaluación de Perfomance

En lo que sigue, se describe el proceso de evaluación de los métodos descriptos, aplicados al caso de sistemas obligatorios k-PMC / opcionales recompensadas con la optimización en los hiperperíodos sucesivos por el método de recocido simulado.

5.1 Generación del conjunto de problemas.

Para la fase experimental, se generaron aleatoriamente alrededor de 1700 sistemas de 10 tareas cada uno y con un hiperperíodo menor a 32000 ranuras, a los efectos de acotar la duración del experimento. Los parámetros de cada uno de los conjuntos de tareas siguen una distribución uniforme. El espacio muestral de los periodos es (10, 20, 30, 40, ..., 600) y el factor de utilización total del subconjunto obligatorio, definido como $\Sigma\ m_i/T_i$, varía en el intervalo [0.13, 0.97].

Si U_m representa el factor de utilización total del subconjunto de tareas obligatorias, el factor de utilización del subconjunto opcional, definido como $U_O = \Sigma\ o_i/T_i$, se calculó como 2-U_m. De este modo, se garantiza que siempre habrá opcionales listas para ser ejecutadas cuando haya ranuras libres.

La selección de los tiempos de ejecución obligatorios y opcionales de las distintas tareas se hace de modo aleatorio sujeto a que se verifiquen las restricciones sobre los factores de utilización y que una tarea sea completamente ejecutable dentro de su período ($m_i+o_i \leq T_i$).

Cada tarea tiene asociada una función recompensa, $f_r(t)$. Esta función recompensa es lineal con una pendiente que fue elegida aleatoriamente en el intervalo [4, 40].

Con los problemas así generados, se realizaron para los métodos propuestos DSM1 y DSM2, tres simulaciones para cada uno durante 50 hiperperíodos y tres para 200 hiperperíodos obteniéndose luego, la recompensa promedio.

5.2 Análisis de resultados

Las curvas (Figuras 1 a 4) muestran, en ordenadas, la relación entre la recompensa total obtenida después de 50 y 200 hiperperíodos para cada uno de los dos métodos DSM y la recompensa obtenida por los métodos DSM manteniendo invariante el valor de los k_i en su máximos. En abscisas se representa el factor de utilización total del subsistema obligatorio, U_m. Para todos los factores de utilización representados en las simulaciones, los dos algoritmos tienen un mejor comportamiento que el utilizado con los k_i invariante. Esta mejora se torna más importante a medida que se incrementa U_m debido a que, en estos casos, se maximiza el beneficio introducido por la mejor administración del escaso tiempo ocioso disponible por parte de los métodos propuestos. Esta ventaja comparativa desaparece con U_m muy próximos a los valores de saturación del sistema debido a que, el subsistema obligatorio, no deja tiempo disponible para la ejecución de tareas opcionales.

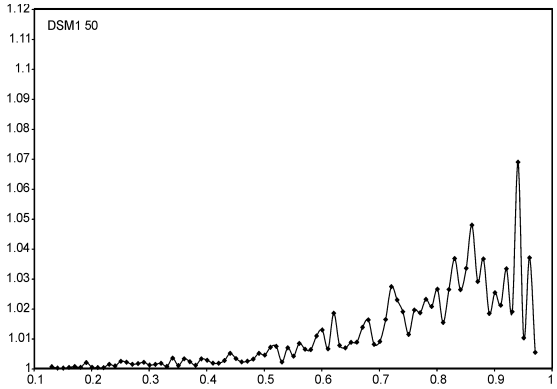

Fig.1 - DSM1 50 Hiperperíodos.

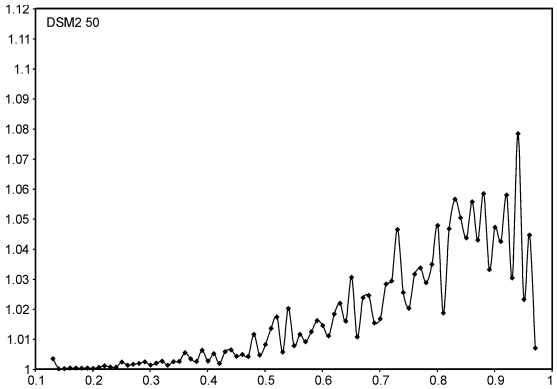

Fig.2 - DSM2 50 Hiperperíodos.

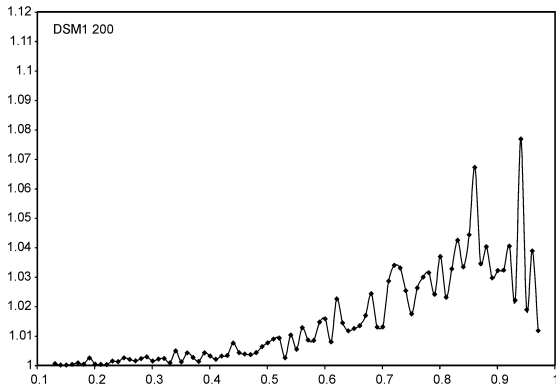

Fig.3 - DSM1 200 Hiperperíodos.

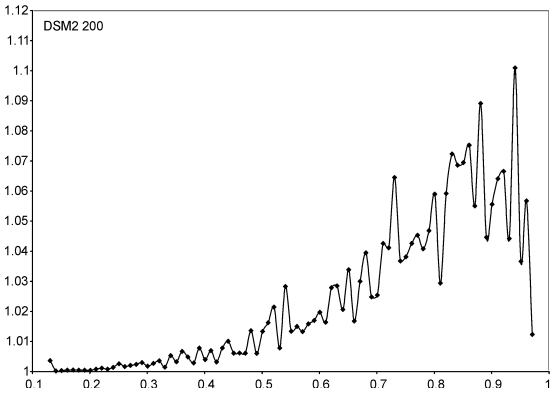

Fig.4 - DSM2 200 Hiperperíodos.

6 Conclusiones

En este trabajo se presenta una forma de optimación, en tiempo de ejecución, para mejorar el desempeño de dos métodos de diagramación de STR compuestos por tareas que tienen una parte obligatoria de tiempo real duro y una opcional. Las subtareas opcionales tienen asociadas funciones recompensa que miden el beneficio que su ejecución brinda al sistema. Si bien no por márgenes muy amplios, los métodos propuestos producen mejoras sobre los resultados obtenidos con otras técnicas además de permitir, su ejecución *on-line* y admitir funciones recompensa no continuamente

diferenciables. Por otro lado, estas ventajas son alcanzables en unos pocos hiperperíodos de optimización. La única sobrecarga que estos métodos imponen es el mantenimiento de los contadores y la carga que impone el algoritmo de recocido simulado.

7 Referencias

[1]. Aydin, H., Audsley, N. C., R. I. Davis and A. Burns, "Mechanism for enhancing the flexibility and utility of hard real-time systems", *Proc. 15th IEEE Rea-Time System Symposium*, pp. 12-21, 1994.

[2]. R. Melhem, D. Mossé and P. Mejía-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks", *IEEE Transactions on Computers,* 50, 2, pp. 111-130, 2001.

[3]. Chung, J. Y., J. W.-S. Liu and K. J. Lin, "Scheduling periodic jobs that allow imprecise results", *IEEE Trans. on Computers*, 19, 9, pp. 1156-1173, 1990.

[4]. Dey, J. K. and J. Kurose, "On line scheduling policies for a class of IRIS (Increasing reward with increasing service) real-time tasks", 46, 7, pp. 802-813, 1986.

[5]. Friedrich, L., J. Stankovic, M. Humphrey, M. Marley and J. Haskins, "A survey of configurable component-based operating systems for embedded applications", *IEEE Computer*, 21, 3, pp. 54-67, 2001.

[6]. Joseph M. and P. Pandya, "Finding response times in a real-time system", *The Computer Journal*, 29, 5, pp. 390-395, 1986.

[7]. Kuo T.-W and A. K. Mok, "Incremental reconfiguration and load adjustment in adaptive real-time systems", *IEEE Transactions on Computers*, 48, 12, pp. 1313-1324, 1997.

[8]. Lin K-J., S. Natarajan and J. W. S. Liu, "Imprecise results: utilizing partial computations in real-time systems", *Proc. 8th IEEE Real-Time Systems Symposium*, pp. 210-217, 1987.

[9]. Liu C. L. and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real-time environment", *J. ACM*, 20, 1, pp. 46-61, 1973.

[10]. Liu J. W.-S, K.-J. Lin, W.-K. Shih, A. C.-S. Yu, C. Chung, Y. Yao and W. Zhao, "Algorithms for scheduling imprecise computations", *Computer*, 24, 5, pp. 58-68, 1991.

[11]. Obenza R., "Rate monotonic analysis for real-time systems", *IEEE Computer*, 26, 3, pp. 73-74, 1993.

[12]. Orozco J., R. Santos, J. Santos and R. Cayssials, "Taking advantage of priority inversions to improve the processing of non-hard real-time tasks in mixed systems", *Proc. WIP 21st IEEE Real-Time Systems Symposium,* pp. 13-16, 2000.

[13]. Santos, J. and J. Orozco, "Rate monotonic scheduling in hard real-time systems", *Information ProcessingLetters*, 48, pp. 39-45, 1993.

[14]. Santos, R. M., J.Urriza, J. Santos and J. Orozco, "Heuristic use of singularities for on-line scheduling of real-time mandatory/reward-based optional systems", *Proc. 14th Euromicro Comference on Real-Time Systems,* 2002.

[15]. Shih, W.-K and J. W. S Liu, "Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error", *IEEE Transactions on Computers,* 44, 3, pp. 466-471, 1995.

[16]. Stankovic, J. and K. Ramamritham, "Advances in Real-Time Systems", *IEEE Computer Society Press*, pp. 1-16, ISBN 0-8186-3792-7, 1992.

[17]. Surmann, H. and A. Morales, "A five layer sensor architecture for autonomous robots in indoor environments", *Proc International Ssymposium on Robotics and Automation, ISRA 2000,* Mexico, pp. 533-538, 2000.