Diagramación *on-line* de sistemas de tareas de tiempo real periódicas mandatorias duras/opcionales basadas en recompensas con factor de depreciación

R. M. Santos, J. Urriza, J. Santos and J. Orozco

Universidad Nacional del Sur/CONICET
8000 Bahía Blanca, Argentina
(iesantos@criba.edu.ar)

Resumen. El trabajo aborda el problema de la diagramación on line de sistemas de tareas de tiempo real periódicas compuestas por una parte mandatoria dura y una parte opcional de cuya ejecución deriva una recompensa. Se consideran funciones recompensa con un factor de depreciación que disminuye el valor de la función a medida que la ejecución de la parte opcional se aleja de la finalización de su correspondiente mandatoria. La maximización del beneficio es un problema NP-duro cuya resolución on-line es impracticable. Se presentan cuatro algoritmos que diagraman las partes opcionales a partir de ciertos instantes singulares en la ejecución del sistema. Los métodos, llamados en forma genérica SH (Singularidades/Heurísticas), utilizan una heurística para resolver localmente la elección de la tarea optativa que más beneficio brinda al sistema. Al contrario de métodos propuestos previamente, los SH no requieren que la función recompensa sea continuamente diferenciable. La única exigencia es que sea computable, incluída la depreciación, en el instante en que va a ser usada. Se evalúa la performance de los métodos propuestos vs Best Incremental Result, usado como base de comparación en trabajos similares.

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 puede perderse ningún vencimiento, el sistema se denomina *duro,* en oposición a los *blandos*, en los cuales se toleran algunas pérdidas. Cuando el sistema cumple con todas las constricciones de tiempo, se dice que es *diagramable*. La diagramabilidad puede ser estudiada en distintos aspectos como, por ejemplo, sistemas operativos, lenguajes de programación, arquitecturas, tolerancia a las fallas, arquitecturas, algoritmos de diagramación, etc. [16].

En los últimos años han encontrado un nicho importante de aplicaciones tecnológicas ciertos sistemas de tiempo real en los cuales las tareas están compuestas de una parte o subtarea dura, de ejecución *mandatoria*, y otra parte o subtarea de

ejecución *opcional*, que en sucesivas iteraciones disminuyen el error del resultado obtenido por la parte mandatoria [3, 8, 10, 15]. En algunos de estos sistemas se asocia una cierta recompensa a la ejecución de opcionales. Las funciones recompensa del mundo real son no decrecientes (no hay recompensas negativas) y, por lo tanto, lineales o cóncavas (*v.g.* exponenciales o logarítmicas). En este último caso, aunque la recompensa total obtenida se incrementa al asignar nuevas ranuras de tiempo a la tarea opcional, el rendimiento es decreciente [2, 4, 14].

En este trabajo se presentan métodos de diagramación para sistemas mandatorios duros/opcionales recompensados, operables *on-line*, en los cuales, además, la función recompensa experimenta una depreciación a partir del momento en que termina de ejecutarse la parte mandatoria.

Esta clase de sistemas es adecuada para la implementación de controladores dedicados a ciertas aplicaciones. El factor de depreciación que se agrega a las funciones recompensa busca evitar que se pierda tiempo refinando un cálculo cuya parte principal está próxima a cambiar por encontrarse ya cercana una nueva generación de la parte mandatoria. Maximizar la recompensa que brinda la ejecución de las opcionales es un problema NP-duro [4].

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

El resto del trabajo se organiza de la siguiente manera: 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 y se presenta el concepto de singularidad; en la Sección 4 se combinan los métodos de singularidades con heurísticas apropiadas para la diagramación de sistemas mandatorios / opcionales con factor de depreciación; en la Sección 5 se evalúa la performance de estos métodos y se la compara con la producida por el método *Best Incremental Return* (BIR), a menudo tomado como base de comparación; finalmente en la Sección 6 se extraen 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 cuales 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 alto 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 mandatoria (con un tiempo de servicio mínimo) y una opcional. Aunque pueden incluirse tareas mandatorias, 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 la parte mandatoria,.

En [3] se proponen distintos esquemas que ejecutan primero la parte mandatoria de todo el sistema y luego seleccionan la parte opcional de acuerdo a diferentes criterios (*v.g.* menor tiempo al vencimiento, menor período, etc.). Las ranuras vacías que aparecen naturalmente en el sistema cuando no hay partes mandatorias pendientes, se llaman ranuras de *background*. Los esquemas fueron comparados en [2] y se encontró que los mejores resultados se obtienen asignando cada ranura vacía a la opcional que mayor recompensa brinde en ese momento. El método se llama *Best Incremental Return* (BIR) y es utilizado como base de comparación.

En [2], se presentó un método *off-line* para distribuir las opcionales en las ranuras vacías del sistema. Utiliza programación lineal para optimizar la función recompensa total del sistema y exige para ello que las funciones individuales 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 mandatoria, número que queda fijo y hace el sistema sumamente rígido. Además el método provee soluciones no sólo restringidas en ese aspecto sino que además las partes mandatorias no pueden ser diagramadas por la disciplina de *Períodos Monotónicos Crecientes* (PMC), excepto en el caso en que los periodos sean armónicos. Este es un inconveniente grave si se tiene en cuenta que PMC constituye una norma *de facto*,.por lo menos en USA, y que, generalmente, los sistemas del mundo real no son armónicos.

Los métodos propuestos en este trabajo no tienen ninguno de estos inconvenientes: El subsistema mandatorio es diagramable por PMC y las funciones recompensa no necesariamente tienen que ser continuamente diferenciables. De hecho, puede utilizarse cualquier función que sea computable en la ranura asignada. Los métodos pueden ser usados *on-line*, tienen una baja carga protocolar y un comportamiento superior al BIR. Además, en el caso de una reducción en la parte mandatoria de una tarea, el tiempo libre puede ser aprovechado para ejecutar opcionales de esa tarea o de cualquier otra. La función recompensa por otro lado puede variar en el tiempo, cambiando de acuerdo a variaciones externas del ambiente, constituyendo así un sistema *adaptivo* al mismo [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 quebrar 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. 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).

4. SISTEMAS MANDATORIOS 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 mandatorias 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 mandatoria 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 mandatorio. *M* denota el mínimo común múltiplo de los periodos de las *n* tareas y se denomina hiperperiodo 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 es, por lo tanto, el número de ranuras necesarias para procesar el subsistema mandatorio 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 hiperperiodo. Si $M - W_n(M) < \sum O_i$, el sistema es sobresaturado y únicamente una porción de las opcionales podrá ser ejecutada. De hecho, el valor máximo de recompensa que puede ser alcanzado en un sistema durante un hiperperiodo, es función del número de ranuras vacías disponibles para la ejecución de opcionales. Este tiempo ocioso se conoce como *slack-time* y la manera más sencilla de aprovecharlo es usarlo cuando aparece pasivamente (*background*). Por el contrario, los algoritmos presentados en este trabajo para atacar el problema de la asignación de ranuras vacías a subtareas opcionales recompensadas, pertenecen a los métodos activos que redistribuyen el slack a lo largo del hiperperiodo, adelantando las ranuras background.

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 a independientes periódicas apropiables a ser diagramado con partes mandatorias PMC y opcionales recompensadas. Éstas últimas deben ser ejecutadas antes de que se vuelvan a generar sus partes mandatorias, 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 mandatorio es k-PMC diagramable si:

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

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

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

En [12] se demostró formalmente que si el subsistema mandatorio es k-PMC diagramable, k ranuras en un intervalo [s, (s+k-1)] pueden usarse para ejecutar tareas del subsistema opcional. El método DSM refina al anterior al generar no sólo la cota inferior sino tantas ranuras, k_i, como cada subtarea M_i admita.

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 mandatarios k-PMC/opcionales recompensadas. Los métodos, llamados genéricamente SH, serán notados DSS1, DSM1, DSS2 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 opcionales de alto valor de recompensa están asociados a mandatorias de baja prioridad y viceversa. En esos

casos, avanzar la ejecución de las opcionales puede producir resultados adversos si la conclusión de una mandatoria de alta prioridad habilita la ejecución de una opcional de baja recompensa, inhabilitando una ejecución posterior de una de mayor recompensa asociada a una mandatoria de menor prioridad.

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

La segunda heurística en cambio, utiliza el avance del *slack-time* para ejecutar tanto mandatorias (aun quebrando el ordenamiento por PMC) como opcionales. Esto lo hace para ejecutar lo más pronto posible las partes mandatorias asociadas a opcionales de recompensa alta.

En lo que sigue se describen los cuatro algoritmos paso por paso. DSS1 y DSS2 se implementan mediante un solo contador, AC cuyo contenido será denominado *AC*. Los pasos para DSS1 son:

1) $AC = k$ en $t=s$.

2) *If* $AC \neq 0$ y no hay pendientes de ejecución mandatorias con partes opcionales que posean una recompensa mayor a la de la opcional que se está por ejecutar, *then* se ejecutará la opcional, *else* se sigue el ordenamiento PMC

3) *AC* se decrementa en una unidad por cada ranura asignada a una parte opcional.

Los pasos de DSS2 son:

1) $AC = k$ en $t=s$.

2) *If* $AC \neq 0$ y no existe una mandatoria pendiente de ejecución cuya opcional asociada posea una recompensa mayor a la de la opcional que se va a ejecutar, *then* la opcional es ejecutada, *else* la mandatoria asociada a la mayor recompensa es ejecutada aún cuando se viole el ordenamiento PMC.

3) *AC* se decrementa en uno por cada ranura asignada a una parte opcional o cuando se ejecuta una mandatoria violando el ordenamiento PMC.

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. Los pasos de DSM1 son:

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 mandatoria con una parte opcional que posea mayor recompensa que la opcional a ser ejecutada, *then* la 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.

Los pasos para el DSM2 son

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 mandatorias pendientes de ser ejecutadas con partes opcionales asociadas con recompensa mayor a la de la opcional a ser

ejecutada, *then* la opcional es ejecutada, *else* la mandatoria 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 hacerse notar que dado que las ranuras vacías constituyen singularidades, 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 mandatoria o hasta la próxima singularidad. En ese instante sin embargo los contadores son recargados y nuevamente tenemos k ranuras disponibles. La actualización de los contadores es la única sobrecarga que imponen los métodos SH.

5. EJEMPLOS DE APLICACION Y EVALUACIÓN DE PERFORMANCE

Un ejemplo de aplicación es el caso de vehículos autónomos moviéndose en ambientes interiores, por ejemplo edificios de oficinas de varios pisos [17]. Cuando el robot se desplaza por un pasillo debe conservar su mano y, en consecuencia, la resolución de la medida a la pared más próxima (cm), debe ser mejor que a la pared opuesta (dm). Ello se consigue asignando mayor recompensa a la ejecución de opcionales que refinen la primera de las medidas. Cuando el robot se acerca a una intersección T debe incrementar la precisión de la medida de distancia a la pared obstáculo. La función recompensa, en consecuencia, puede tener un cierto valor para distancias superiores a 3 *mts* e incrementar ese valor para mejorar la precisión de las medidas de distancias menores. Por lo tanto, el refinamiento de la medida de distancia es función de la distancia misma y se tiene un sistema adaptivo que varía su comportamiento según la posición del robot en su entorno. Si a esto se suma el hecho de que la importancia del refinamiento disminuye a medida que se aleja de la medida mandatoria ya ejecutada y se acerca a una nueva medida mandatoria, se hace evidente la razón para afectar la función recompensa con un factor de depreciación.

Algo similar sucede en procesamiento dinámico de imágenes. Aunque en general resulte deseable incrementar la definición de una imagen, puede no resultar conveniente hacerlo si fue tomada tiempo atrás y se está a punto de obtener una nueva.

En lo que sigue se describe el proceso de evaluación de los métodos descriptos, aplicados al caso de sistemas mandatorios k-PMC/opcionales recompensadas con funciones depreciadas en función del tiempo. La evaluación se detalla de tal modo que puede ser validada y replicada.

5.1. Generación de los sistemas aleatorios.

Se generaron más de 15000 conjuntos aleatorios de 5 tareas. Los parámetros de cada uno de los conjuntos siguen una distribución uniforme. El espacio muestral de los

periodos es (10, 20, 30, 40, ..., 600), se restringe el tamaño de los hiperperiodos a 32000. El factor de utilización total del subconjunto mandatorio, definido como $\Sigma\ m_i/T_i$, varia en el intervalo [0.06, 0.9]. El factor de utilización total del subconjunto opcional, definido como $\Sigma\ o_i/T_i$, se calculó como 2 menos el factor de utilización total mandatorio. De este modo se garantiza que siempre habrá opcionales listas para ser ejecutadas cuando haya ranuras libres. Los tiempos de ejecución mandatorios y opcionales entre las distintas tareas se hace de modo aleatorio sujeto a que se verifiquen ambos factores de utilización y que se cumpla $m_i+o_i \leq T_i$.

Cada tarea tiene asociada una función recompensa, $f_r(t)$. Se generaron tres tipos de funciones: lineal, exponencial y logarítmica. En los tres casos, en $t=o_i$, ellas alcanzan el mismo valor de recompensa, un número entero que se elige con distribución uniforme en el intervalo [4, 40]. La función recompensa sufre una depreciación a partir del instante en que termina de ejecutarse la mandatoria. La depreciación esta dada por una función $f_d(t-t_f)$, en la que t_f denota la última ranura de ejecución de la mandatoria. La función recompensa depreciada es pues:

$$f_{rd}(t) = f_r(t)\ f_d(t-t_f)$$

En las simulaciones se consideró una depreciación exponencial, $exp\ [-\alpha\ (t-t_f)]$ en la que $\alpha = (\ln a)/T_i$ y a se genera de manera aleatoria. Los resultados se muestran en las Fig. 1 a 4, para DSS1, DSM1, DSS2 y DSM2, respectivamente.

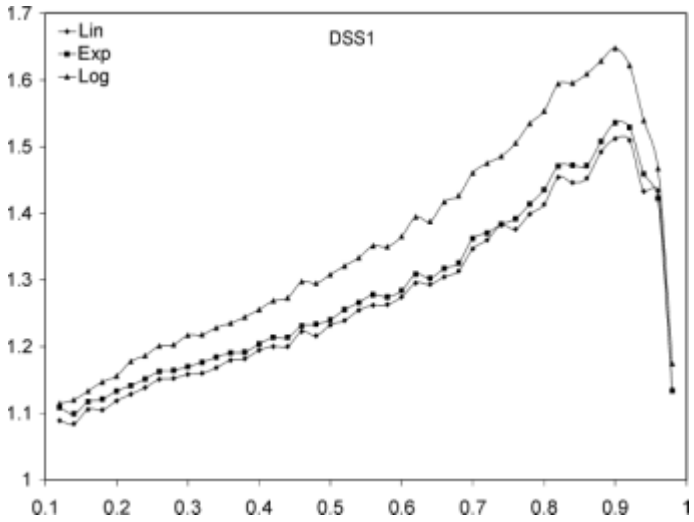

Fig. 1.

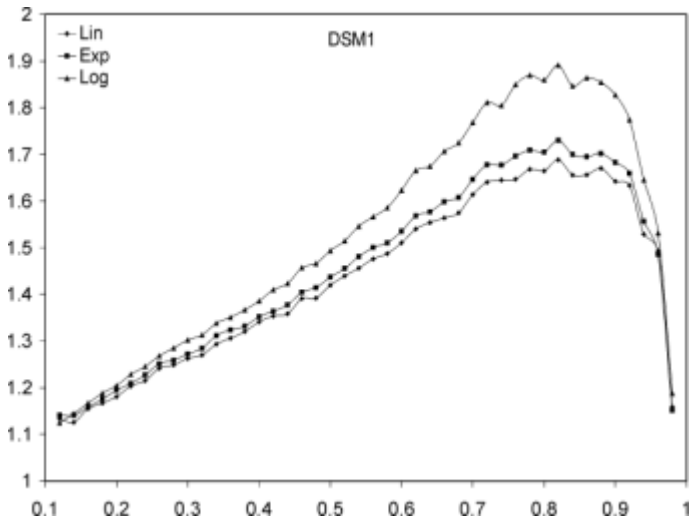
DSM1
Lin
Exp
Log
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Fig. 2.

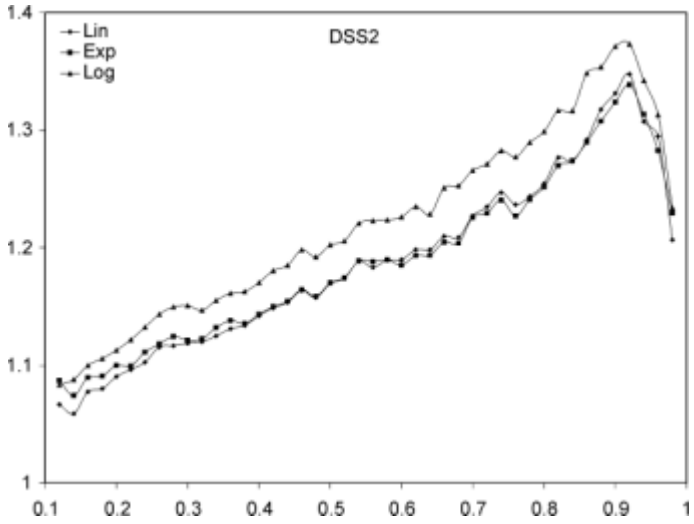
DSS2
Lin
Exp
Log
1.4
1.3
1.2
1.1
1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Fig. 3.

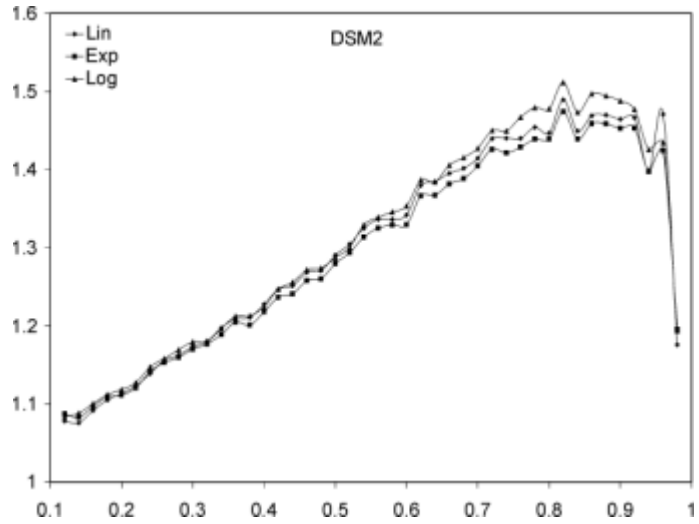

Fig. 4.

5.2. Análisis de resultados

Las curvas muestran, en ordenadas, la relación entre la recompensa total obtenida en un hiperperiodo por cada uno de los cuatro métodos SH y la recompensa obtenida por el método BIR para cada uno de los tres tipos de función. En abcisas se representa el factor de utilización mandatorio total, FU. Para todos los factores de utilización los cuatro algoritmos tienen un mejor comportamiento que el BIR. Lo superan en alrededor de un 10% cuando el FU es pequeño. A partir de allí, la relación aumenta y alcanza un pico cerca de 0.9. Esto se debe a que a medida que se incrementan los tiempos de ejecución de las tareas, las ranuras vacías background tardan más en aparecer y, por lo tanto, se ejecutan en ellas opcionales muy depreciadas. Los métodos SH, al permitir la ejecución de las partes opcionales antes de concluir la ejecución de todas las partes mandatorias pendientes, logran recompensas mayores por estar menos afectadas por el factor de depreciación. Con factores de utilización muy altos, la relación disminuye. Esto se debe a que hay muy pocas ranuras vacías disponibles y los métodos SH no pueden utilizar plenamente su potencialidad.

Los métodos de detección múltiple muestran sistemáticamente mejores resultados que sus contrapartes de detección simple.

6. CONCLUSIONES

En este trabajo se presentaron cuatro métodos diferentes, ejecutables *on-line*, para resolver el problema de la diagramación de STR compuestos por tareas que tienen una

parte mandatoria a ejecutar antes del vencimiento y una opcional que mejora de alguna manera los cálculos realizados por la primera. Las subtareas opcionales tienen asociadas funciones recompensa que miden el beneficio que la ejecución de las mismas brinda al sistema. La función recompensa se deprecia en función de la distancia temporal al término de la ejecución de su mandatoria. Esto busca evitar que se le asigne tiempo a una tarea para refinar un cálculo que está próximo a ser actualizado en su parte fundamental. Ejemplos de estas aplicaciones surgen en el control de vehículos autónomos, procesamiento de imágenes, sistemas de seguimiento predictivos, etc. La optimización de la ganancia es de complejidad NP-dura. En trabajos previos se presentaron métodos de diagramación que adolecen de dos inconvenientes graves: no pueden ser ejecutados *on-line* y requieren que las funciones recompensa sean continuamente diferenciables. Ninguno de ellos contempla, además, la depreciación de la recompensa a medida que se aleja de la terminación de la mandatoria. Los cuatro algoritmos propuestos aquí, que aprovechan el adelantamiento de las ranuras vacías para la ejecución de las partes opcionales, mejoran significativamente la performance de Best Incremental Result, a menudo usado como base de comparación. El único requerimiento que estos métodos imponen es que las funciones sean computables en todo momento y su *overhead* es el mantenimiento de los contadores.

REFERENCIAS

[1] 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] Aydin, H., 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,* (to be published), 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.