Test de Planificabilidad de Bajo Costo Computacional para RM/DM

José M. Urriza[1], Francisco E. Paez[1], Javier D. Orozco[2], Ricardo Cayssials[2,3]

[1] Universidad Nacional de la Patagonia San Juan Bosco, Depto. Informática, Fac. Ingeniería, Sede Puerto Madryn, Argentina

josemurriza@gmail.com; fep@ing.unp.edu.ar

[2] Universidad Nacional del Sur, DIEC, Bahía Blanca, Argentina

jdorozco@gmail.com

[3] Universidad Tecnológica Nacional, Fac., Regional Bahía Blanca

rcayssials@frbb.utn.edu.ar

Resumen. Un sistema de tiempo real planificado por la disciplina de prioridades fijas *Rate Monotonic* (*RM*) o *Deadline Monotonic* (*DM*), necesita de un test de planificabilidad para validar su correcto funcionamiento. Esta validación resulta ineludible en sistemas críticos. Si además, es necesario realizarla en línea, su propio tiempo de ejecución supeditará la factibilidad del sistema, debido al costo temporal que el propio test introduce. Es por ello que, toda mejora que reduzca los tiempos de cómputo del test, es beneficiosa. Este trabajo presenta una mejora a los mecanismos actuales, que reduce el costo computacional temporal de otros métodos desarrollados por los autores, en trabajos previos.

1 Introducción

En los *Sistemas de Tiempo Real* (*STR*) es deseable y hasta necesario que el vencimiento de las tareas se respete. En caso de ser *STR* críticos, su cumplimiento es imperioso para preservar el buen funcionamiento del sistema. Aún si el sistema admitiera pérdidas ocasionales de vencimientos bajo algún tipo de restricción más débil, es necesario realizar un análisis de factibilidad que garantice a priori, que las tareas cumplen con las condiciones temporales que se le imponen.

Por otro lado, la complejidad computacional del análisis de factibilidad, es dependiente de la disciplina utilizada. En disciplinas óptimas, como *EDF* o *LLF*, se sabe que si el *Factor de utilización* (*FU*) es $FU \leq 1$ el *STR* es planificable [1]. Sin embargo, en disciplinas como *Rate Monotonic* (*RM*) o *Deadline Monotonic* (*DM*) se debe realizar un test de planificabilidad de mayor complejidad.

Para estos últimos, se han desarrollados numerosos tests desde los comienzos de la disciplina y continúa el desarrollo de los mismos hasta la actualidad. En éstos, se busca reducir la complejidad espacial y/o temporal de los algoritmos para que puedan ser utilizados en-línea. En otras palabras, reducir el *costo computacional* (*CC*) temporal y espacial del cálculo.

Para *RM* o *DM*, se han presentado numerosos tests de planificabilidad de dos tipos. Existe los tests de *condición suficiente* y los que son de *condición necesaria y suficiente*. Los primeros, en su gran mayoría están basado en cotas [1, 2], que se determinan utilizando distintos parámetros del sistema. Si el *STR* cumple con la cota, se garantiza que es planificable. Si no cumple, no se puede asegurar que el sistema sea planificable. Para los que son de *condición necesaria y suficiente*, la determinación mediante el test, garantiza que el *STR* es o no es planificable.

A continuación, se realiza una introducción a los *STR* y posteriormente, a los test de planificabilidad. Luego en el punto 2 se presentan las distintas mejoras que han tenido los Test para *RM*/*DM*. En el punto 3 se presenta la mejora propuesta. En 4 se presentan los resultados experimentales. Finalmente, en 5 se realizan las conclusiones.

1.1 Introducción a los *STR*

Los *STR* encuentran aplicación en innumerables dispositivos computacionales, desde los de distribución masiva hasta los más complejos y de propósito crítico. Es posible encontrarlos en dispositivos para la industria en aplicaciones de control, instrumentación para la milicia, aviónica, electrodomésticos, computadoras de automóviles, teléfonos celulares, etc.

Los diseñadores de estos dispositivos, utilizan herramientas, que permiten validar que el *STR* funcione de manera correcta y así garantizar que estos cumplen con el objetivo para el cual fueron creados. Es común aseverar en la disciplina que: “En los *STR* 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” ([3]).

El vencimiento es una constricción temporal y es un parámetro que es propio de cada tarea del *STR*. Debido a esto, los *STR* se clasifican en *duros* si no toleran perdidas en los vencimientos, en *blandos* si toleran algunas pérdidas y en *firmes,* si el número de perdidas está acotado.

En [1] se definió uno de los primeros modelos de tareas, a partir del cual se desarrollaron otros modelos y métodos, para estos *sistemas dinámicos, no-lineales, discretos y determinísticos* ([4]). Tener un modelo permite analizar la evolución temporal de un *STR*, como los presentados en [5, 6].

Por otro lado, en [1] se demostró que el *peor estado de carga* de un sistema multi usuario – mono recurso (específicamente multitarea – monoprocesador), es aquel en el cual todas las tareas periódicas requieren ser atendidas en el mismo instante, y se denomina *instante crítico*. Cuando el sistema comienza desde este instante y en su evolución temporal cumple con todas las constricciones temporales impuestas en la primera instancia de cada tarea, se dice que el *STR* es planificable. Por otro lado, se considera que el periodo de una tarea se mantiene invariante, a lo largo del hiperperíodo y por lo tanto, se mantiene constante el tiempo interarribo.

A continuación, en la *Tabla* 1, se detallan los parámetros de una tarea para un modelo básico, el cual es similar al presentado en los trabajos [1, 6] entre otros.

Tabla 1. Parámetros Básicos de una Tarea.

Tiempo de ejecución (C_i)	*Máximo tiempo que podría tomar la ejecución de una tarea, que se define como el peor caso de tiempo de ejecución* (*WCET*).
Período (T_i)	*Intervalo de tiempo entre arribos de una tarea periódica.*
Vencimiento (D_i)	*Instante de tiempo máximo, relativo a la instanciación de la tarea, en que la misma debe completar su ejecución.*
Tiempo de bloqueo (B_i)	*Máximo tiempo en que una tarea podría esperar por tareas de menor prioridad con las que comparte recursos.*
Factor de Utilización (FU_i)	*Factor de Utilización de la tarea i,* $FU_i = C_i / T_i$

En lo siguiente, para especificar un sistema de n tareas, se utilizará la siguiente notación para el modelo básico $S(n) = \{(C_1, T_1, D_1),\ldots, (C_n, T_n, D_n)\}$.

1.2 Introducción a los Test de Planificabilidad para RM/DM

En 1986, Joseph y Pandya ([5]), presentan un test de planificabilidad para las disciplinas de prioridades fijas *RM/DM*, que calcula el *peor caso de tiempo de respuesta* (*WCRT: Worst Case Response Time*) de cada tarea del sistema. Tests similares fueron presentados en [7, 8, 9, 10, 11, 12, 13]

Un *punto fijo* (*PF*), en una función de un *sistema dinámico*, representa un instante en donde el tiempo es numéricamente igual a la función ($t = f(t)$). En [5], se presentó un modelo discreto para evolucionar un *sistema dinámico* por prioridades fijas, de la *tarea* 1 hasta la n. El test, parte de una condición inicial, que es el *instante crítico*, hasta una condición final, la cual es encontrar un *PF* antes que la evolución de las iteraciones supere el vencimiento de la tarea analizada. Si todas las tareas cumplen esta condición, el *STR* es planificable. Si la primera instancia de alguna tarea supera el vencimiento, la misma es no planificable y tampoco el *STR*. Por lo tanto, este test es de *condición necesaria y suficiente* para determinar la planificabilidad de un *STR* bajo las disciplinas *RM, DM* o cualquier otro ordenamiento por prioridades fijas.

La notación utilizada será, R_i para el instante donde ocurre el *WCRT*, t^q para indicar el instante t de la iteración q y t^{q+1} para indicar el resultado de la iteración q. El proceso iterativo dará un $R_i = t^{q+1} = t^q \leq D_i$ en caso de ser planificable la tarea i o $t^{q+1} > D_i$ para el caso de que no sea planificable. La función es la siguiente:

$$t^{q+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t^q}{T_j} \right\rceil C_j \tag{1}$$

2 Mejoras Realizadas al Test de Planificabilidad para *RM*/*DM*

2.1 RTA - Semilla de Comienzo

En 1998, Sjodin y Hansson [8], introducen una mejora al algoritmo de [5], acercando la semilla de inicio del proceso iterativo t_i^0 al *PF*. De esta manera disminuye el intervalo de inspección donde debe iterar el algoritmo y, en consecuencia, se reduce el *CC* del cálculo. La mejora introducida consiste en comenzar la iteración donde se encontró el *WCRT* de la tarea *i*-1, más el tiempo de ejecución de la tarea *i*. En el modelo simple, $t_i^0 = R_{i-1} + C_i$, para el modelo con bloqueos, $t_i^0 = R_{i-1} - B_{i-1} + C_i + B_i$.

2.2 RTA2 - Mejora al algoritmo

A continuación, se presenta el análisis de la traza generada en [5] y la mejora propuesta en [14]. Se supone un *STR* S(n) = {(C_1, T_1, D_1),..., (C_n, T_n, D_n)} planificable y por lo tanto, con un *PF* en el intervalo $(t_i^0, D_i]$. Cuando se desarrolla el método de [5] a una tarea *i*, con $i \in 1 < i \leq n$ y semilla $t_i^0 = R_{i-1} + C_i$, en la traza ocurrirán m+1 iteraciones antes de llegar al primer *PF*, con $m \geq 0$. En la *Tabla* 2, se realiza la traza genérica.

Tabla 2. Traza de cálculo genérico.

q	$f(t^q)$	q	$f(t^q)$
0	$t^1 = \left\lceil \frac{t^0}{T_1} \right\rceil C_1 + \left\lceil \frac{t^0}{T_2} \right\rceil C_2 + ... + \left\lceil \frac{t^0}{T_{i-1}} \right\rceil C_{i-1} + C_i$	1	$t^2 = \left\lceil \frac{t^1}{T_1} \right\rceil C_1 + \left\lceil \frac{t^1}{T_2} \right\rceil C_2 + ... + \left\lceil \frac{t^1}{T_{i-1}} \right\rceil C_{i-1} + C_i$
....	.....	...	.....
m-1	$t^m = \left\lceil \frac{t^{m-1}}{T_1} \right\rceil C_1 + \left\lceil \frac{t^{m-1}}{T_2} \right\rceil C_2 + ... + \left\lceil \frac{t^{m-1}}{T_{i-1}} \right\rceil C_{i-1} + C_i$	m	$t^{m+1} = \left\lceil \frac{t^m}{T_1} \right\rceil C_1 + \left\lceil \frac{t^m}{T_2} \right\rceil C_2 + ... + \left\lceil \frac{t^m}{T_{i-1}} \right\rceil C_{i-1} + C_i$

Alcanzado el *PF* en t^m, el resultado de la iteración m-1 (t^m) es igual al resultado de la iteración m (t^{m+1}).

Si se analiza la función, el cálculo de $\lceil t^q / T_j \rceil$ determina la cantidad de instanciaciones de la tarea *j* en el intervalo $[0, t^q)$. Luego $\lceil t^q / T_j \rceil C_j$ indica el tiempo de ejecución acumulado de todas las instancias de la tarea *j* en el intervalo [0,*t*), denominado *carga*

de trabajo de la tarea j ($W_j(t)$). Como se observa, $W_j(t)$ es el *invariante* del algoritmo y de aquí en adelante, se indica a la carga de trabajo de la tarea *j,* en el instante *t,* de la iteración *q*, como se presenta en la ecuación (2).

$$A_j^q = \left\lceil \frac{t^q}{T_j} \right\rceil C_j \quad \text{con} \quad 0 \le q \le m \quad \text{y} \quad 1 \le j \le i-1 \tag{2}$$

El valor de A_j^q tiene un intervalo de validez, donde no cambia su valor. El mismo comienza cuando arriba la instancia *k*, hasta que arriba una nueva instancia *k*+1. Para una tarea con periodo T_j , el intervalo es $(kT_j,(k+1)T_j]$. En consecuencia, en todo instante *t* perteneciente a este intervalo, se obtendrá el mismo valor A_j^q .

El algoritmo se detiene cuando se cumple $t^{m+1} = t^m$, que es un *PF.* Esto implica que los pares A_j^{m-1}, A_j^m con $1 \le j \le i-1$ son todos iguales (ecuación (3)).

$$t^m = A_1^{m-1} + A_2^{m-1} + \ldots + A_{i-1}^{m-1} + C_i = A_1^m + A_2^m + \ldots + A_{i-1}^m + C_i = t^{m+1} \tag{3}$$

Por otro lado, reemplazando t^{q-1} en cada iteración, cualquier A_j^q queda:

$$A_j^q = \left\lceil \frac{A_1^{q-1} + A_2^{q-1} + \ldots + A_{i-1}^{q-1} + C_i}{T_j} \right\rceil C_j \text{ con } 1 \le j \le i-1 \tag{4}$$

Si se iguala a cero las iteraciones *q* y *q*+1 en (1) y se despeja t^{q+1} , se tiene:

$$t^{q+1} - C_i - \sum_{j=1}^{i-1} A_j^q = t^q - C_i - \sum_{j=1}^{i-1} A_j^{q-1} \Rightarrow t^{q+1} = t^q + \sum_{j=1}^{i-1} A_j^q - A_j^{q-1} \tag{5}$$

Como se puede observar en (5), el avance del algoritmo hacia el *PF*, depende de la sumatoria de las diferencias de los A_j^q , entre las iteraciones. Además, si existe algún par $A_j^q - A_j^{q-1} > 0$ entonces $t^{q+1} \neq t^q$ y se realizará al menos una iteración más con la nueva semilla t^{q+1} para encontrar el *PF*. Con los algoritmos presentados en [5, 8], un incremento en A_j^q no afecta al cálculo de los siguientes $A_{j+1}^q, \ldots, A_{i-1}^q$ hasta la próxima iteración, dado que siempre se itera con t^q (ver (4)).

En [14], se probó que es posible incrementar t^q , cada vez que se encuentra que $A_j^q - A_j^{q-1} > 0$ con $t^{q+} = t^q + A_j^q - A_j^{q-1}$ y no esperar a la siguiente iteración. De esta manera, se puede calcular los siguientes $A_{j+1}^q, \ldots, A_{i-1}^q$ con la nueva semilla t^{q+} . Esto genera una velocidad de convergencia mayor o a lo sumo igual. El *CC* introducido es almacenar cada valor del A_j^{q-1} , lo cual posee un costo lineal. Para ahorrar tiempo y costo, al momento de inicialización del arreglo, $A_j^0 = C_j$.

2.3 RTA3 - Reducción en el Número de Cálculos de Invariantes.

El cálculo de A_j^q , es lo que incrementa notablemente el *CC* temporal del algoritmo, por las operaciones de punto flotante y comparaciones que debe realizar. La mejora propuesta en RTA3 [15], disminuye el número de cálculos de A_j^q , lo cual reduce el costo notablemente. Además, suma dos teoremas que permiten reducir el número de iteraciones y, bajo ciertas condiciones de los A_j^q , no realizarlas.

Como fue presentado en el punto anterior, cada valor de A_j^q tiene un intervalo de validez. En consecuencia, si t^q pertenece al intervalo de validez de A_j^{q-1}, el cálculo resultará en el mismo valor, $A_j^q = A_j^{q-1}$. Al extremo superior de aquí en adelante se lo denominará I_j^q, que se almacenará en un arreglo, inicializado con $I_j^q = T_j$ para todo $1 \le j \le n$ y se deberá actualizar al cambiar A_j^q.

La mejora que se propone en RTA3, es realizar el cálculo de A_j^q y I_j^q, si y solo si, $t^q > I_j^{q-1}$, dado que si $t^q \le I_j^{q-1}$ se sabe qué $A_j^q = A_j^{q-1}$ y $I_j^q = I_j^{q-1}$.

El siguiente teorema, fue probado en [15] y evita la última iteración:

Teorema 1:

Sea un STR de n tareas, si al finalizar una iteración se encuentra que todos los I_j^q con $1 \le j \le i$, son mayores o iguales a t^{q+1}, el algoritmo ya se encuentra en un PF.

El *Teorema* 2, fue probado en [16] tomando como semilla $t_i^0 = R_{i-1} + C_i$ de [8] y es una mejora que se agrega al RTA3 original:

Teorema 2:

Sea un STR de n tareas, si la semilla de inicio del Test de la tarea i, t_i^0, es menor que todos I_j^0 con $1 \le j \le i-1$, entonces t_i^0 es un PF para la tarea i y es planificable si cumple con ser menor al vencimiento.

Este teorema permite ahorrar sucesivas iteraciones y cálculos, al probar que la primera instancia de la tarea *i* se ejecuta sin ser apropiada, luego de que finaliza la primera instancia de la tarea *i*-1.

Por otro lado, se desprende del *Teorema 2*, que los A_{i-1}^q y I_{i-1}^q, con $1 \le j \le i-1$, utilizados en el cálculo de la tarea *i*-1, son válidos para la tarea *i*, dado que las iteraciones continúan desde $t_i^0 = R_{i-1} + C_i$. Esto contribuye a no incrementar el *CC* del algoritmo.

Antes de proseguir, es importante presentar una notación que permita identificar los cambios introducidos en el método. De aquí en adelante, se colocará en la parte superior del techo los valores A_j^q y I_j^q, separados por ";". Se indica, como superíndice de I_j^q, el número de tarea. Además, en la parte inferior, estará el nuevo t^{q+}. Cuando los $A_j^q \neq A_j^{q-1}$, se presenta el techo para realizar el cálculo (6) y cuando sea $A_j^q = A_j^{q-1}$ (7), simplemente se presenta el valor de A_j^q:

$$t^{q+1} = \sum_{j=1}^{i} \overset{A_j^q ; I_j^{qTarea}}{\left\lceil \frac{t^q}{T_j} \right\rceil C_j} \Bigg|_{t^{q+} = t^q + A_j^q - A_j^{q-1}}^{A_j^q \neq A_j^{q-1}} \qquad (6) \qquad t^{q+1} = \sum_{j=1}^{i} \overset{A_j^q ; I_j^{qTarea}}{A_j^q(t^q)} \Bigg|_{t^{q+} = t^q + A_j^q - A_j^{q-1}}^{A_j^q = A_j^{q-1}} \qquad (7)$$

En la *Tabla* 4 se presenta un ejemplo de RTA3, con S(10)={(8, 126, 126), (11, 206, 206), (116, 340, 340), (32, 436, 436), (1, 532, 532), (53, 742, 742), (18, 817,

817), (26, 869, 869), (64, 873, 873), (18, 906, 906)}. Si bien parece de una resolución compleja y larga, con RTA3 resulta sencilla. Cada *PF* que se encuentre por los teoremas 1 o 2, se indicarán como *T1* o *T2*.

Tabla 3. Ejemplo RTA3 en orden de prioridad.

τ	q	t^q	$f(t^q)$
1	0	0	$C_1 < D_1 => R_1 = 8$
2	0	8	$R_2 = 19$ *T2*
3	0	135	$t_3^1 = \left\lceil \frac{135}{126} \right\rceil^{16;252^1}_{143} 8 + \underset{143}{\overset{11;206^2}{11}} + \underset{143}{\overset{116;340^3}{116}} = 143 \quad R_3 = 143 \quad T1$
4	0	175	$R_4 = 175$ *T2*
5	0	176	$R_5 = 176$ *T2*
6	0	229	$t_6^1 = \underset{229}{\overset{16;252^1}{16}} + \left\lceil \frac{229}{206} \right\rceil^{22;412^2}_{240} 11 + \underset{240}{\overset{116;340^3}{116}} + \underset{240}{\overset{32;436^4}{32}} + \underset{240}{\overset{1;532^5}{1}} + \underset{240}{\overset{53;741^6}{53}} = 240 \quad R_6 = 240 \quad T1$
7	0	258	$t_7^1 = \left\lceil \frac{258}{126} \right\rceil^{24;378^1}_{266} 8 + \underset{266}{\overset{22;412^2}{22}} + \underset{266}{\overset{116;340^3}{116}} + \underset{266}{\overset{32;436^4}{32}} + \underset{266}{\overset{1;532^5}{1}} + \underset{266}{\overset{53;741^6}{53}} + \underset{266}{\overset{18;817^7}{18}} = 266 \quad R_7 = 266 \quad T1$
8	0	292	$R_8 = 292$ *T2*
9	0	356	$t_9^1 = \underset{356}{\overset{24;378^1}{24}} + \underset{356}{\overset{22;412^2}{22}} + \left\lceil \frac{356}{340} \right\rceil^{232;680^3}_{472} 116 + \left\lceil \frac{472}{436} \right\rceil^{64;872^4}_{504} 32 + \underset{504}{\overset{1;532^5}{1}} + \underset{504}{\overset{53;742^6}{53}} + \underset{504}{\overset{18;817^7}{18}} + \underset{504}{\overset{26;869^8}{26}} = 504$
	1	504	$t_9^2 = \left\lceil \frac{504}{126} \right\rceil^{32;504^1}_{512} 8 + \left\lceil \frac{512}{206} \right\rceil^{33;618^2}_{523} 11 + \underset{523}{\overset{232;680^3}{232}} + \underset{523}{\overset{64;872^4}{64}} + \underset{523}{\overset{1;532^5}{1}} + \underset{523}{\overset{53;742^6}{53}} + \underset{523}{\overset{18;817^7}{18}} + \underset{523}{\overset{26;869^8}{26}} = 523$
	2	523	$t_9^3 = \left\lceil \frac{523}{126} \right\rceil^{40;630^1}_{531} 8 + \underset{531}{\overset{33;618^2}{33}} + \underset{531}{\overset{232;680^3}{232}} + \underset{531}{\overset{64;872^4}{64}} + \underset{531}{\overset{1;532^5}{1}} + \underset{531}{\overset{53;742^6}{53}} + \underset{531}{\overset{18;817^7}{18}} + \underset{531}{\overset{26;869^8}{26}} = 531 \quad R_9 = 531 \quad T1$
10	0	549	$t_{10}^1 = \underset{549}{\overset{40;630^1}{40}} + \underset{549}{\overset{33;618^2}{33}} + \underset{549}{\overset{232;680^3}{232}} + \underset{549}{\overset{64;872^4}{64}} + \left\lceil \frac{549}{532} \right\rceil^{1;1064^5}_{550} 1 + \underset{550}{\overset{53;742^6}{53}} + \underset{550}{\overset{18;817^7}{18}} + \underset{550}{\overset{26;869^8}{26}} = 550 \quad R_{10} = 550 \quad T1$

2.4 RTA4 - Reducción de Iteraciones e Invariantes.

El método RTA4 propuesto en [16], funciona de manera similar a RTA3. Sin embargo, reduce aún más la cantidad de invariantes e iteraciones necesarias, si se presentan algunos casos especiales en la iteración de RTA3.

A continuación, con un ejemplo, se presenta lo que ocurre, con los A_j^q, I_j^q, y el t^{q+}, en una iteración con el método RTA3. La ecuación (8), es extraída de la *Tabla* 3, cuando se calcula la *tarea* 9 en la *iteración* 1. Se puede observar que $t^{1+} = 512$ y $I_1^1 = 504$, pero al ser $t^{1+} > I_1^1$, se sabe que en la siguiente iteración $A_1^2 \neq A_1^1$ y que el valor de la semilla para la *iteración* 2 será $t^2 \geq 512$ ¿Por qué esperar a la siguiente iteración?

Es posible iterar sobre el mismo A_1^1, realizando nuevamente el cálculo, sin avanzar al siguiente termino. Esto dará como resultado un nuevo valor de A_1^1, que dará $t^{1+}=520$. En el cálculo de A_2^1, con $t^{1+}=520$ dará el valor del *PF*, con una iteración menos para la tarea 9 (ver ecuación. (9)).

$$\left\lceil \frac{503}{126} \right\rceil_{512}^{32;504^1} 8+... \qquad (8) \qquad \left(\left\lceil \frac{503}{126} \right\rceil_{512}^{32;504^1} 8; \left(\left\lceil \frac{512}{126} \right\rceil_{520}^{40;630^1} 8 \right) \right) + \left\lceil \frac{520}{206} \right\rceil_{531}^{33;618^2} 11+... \qquad (9)$$

Además, en [16], se encontró que es posible para un mismo A_j^q, que se repita numerosas veces $t^{q+} > I_j^q$. Para ejemplificarlo, se pude observar en la ecuación (10), de un *STR* similar al presentado, que no se exhibe por tener una resolución más extensa.

$$\left(\left\lceil \frac{572}{35} \right\rceil_{647}^{255;595^1} 15; \left(\left\lceil \frac{647}{35} \right\rceil_{677}^{285;665^1} 15 \right); \left(\left(\left\lceil \frac{677}{35} \right\rceil_{692}^{300;700^1} 15 \right) \right) \right) + ... \qquad (10)$$

En [16] se encontró que es posible obtener ese resultado de manera analítica, al ser el cálculo del techo de una sola tarea, sin la necesidad de realizar sucesivas iteraciones sobre A_j^q, cuando $A_j^q \neq A_j^{q-1}$ y $t^{q+} > I_j^q$ (Ecuaciones (11) y (12)).

Por otro lado, el resultado de la sumatoria siempre es igual al obtenido por t^{q+1}, por lo cual se propone simplemente realizar la iteración sobre t^q, comenzando con la semilla t^0. Además, se realizan los incrementos de t^{q+1}, desde la tarea de menor prioridad a la de mayor prioridad. En la *Tabla* 4, se realiza el test del *STR* antes presentado con RTA4. Con fines didácticos, se indican aún los $A_j^q{*} - A_j^{q-1}{*} = 0$, como ayuda para seguir la traza.

$$t^{q+} = t^q + \sum_{j=1}^{t^{q+}} {}^{i-1} A_j^q{*} - A_j^{q-1}{*} \qquad (11) \qquad A_j^q{*} = \left\lceil \frac{t^q - A_j^{q-1}{*}}{T_j - C_j} \right\rceil_{A_j^q{*};I_j^q\ Tarea} C_j \qquad (12)$$

3 RTA5 - Una Mejora a RTA4

Es importante resaltar que, toda mejora que reduzca *CC* de un algoritmo utilizado en línea, permite aprovechar el tiempo ganado en otras funciones. De un estudio de las trazas producidas, se ha concluido que *CC* de los métodos presentados es sensible al orden del cálculo de los invariantes, variando su número y/o el número de iteraciones, para un mismo *STR*.

En RTA3 y RTA4, se presentó que existe una mejora si el orden se realizaba de la tarea de menor prioridad a la tarea de mayor prioridad. Sin embargo, se ha encontrado que, si se ordena a las tareas por factor de utilización, en promedio, hay una mejora sobre otros ordenamientos. No obstante, se pueden encontrar algunos ejemplos que con otros ordenamientos que mejoran levemente el *CC* con respecto a RTA5.

Esta mejora, sólo introduce un costo lineal por tarea, dado que solo es necesario encontrar el lugar de la tarea por FU_i dentro del conjunto de tareas y es posible realizarla a medida que el método se ejecuta.

Tabla 4. Ejemplo RTA4 orden de menor a mayor prioridad.

τ	q	t^q	$f(t^q)$
1	0	0	$C_1 < D_1 => R_1 = 8$
2	0	8	$R_2 = 19\ T2$
3	0	135	$135+\underset{135}{\overset{11;206^2}{11}}-11;135+\left\lceil\frac{135-8}{126-8}\right\rceil_{143}^{16;252^1}8-8\ \ R_3=143\ \ T1\ \ R_3 = 143\ T1$
4	0	175	$R_4 = 175\ T2$
5	0	176	$R_5 = 176\ T2$
6	0	229	$229+\underset{229}{\overset{1;532^5}{1}}-1;240+\underset{229}{\overset{32;436^4}{32}}-32;240+\underset{229}{\overset{116;340^3}{116}}-116;229+\left\lceil\frac{229-11}{206-11}\right\rceil_{240}^{22;412^2}11-11;240+\underset{240}{\overset{16;252^1}{16}}-16$ $R_6=240\ \ T1$
7	0	258	$258+\underset{258}{\overset{53;742^6}{18}}-18;258+\underset{258}{\overset{1;532^5}{1}}-1;258+\underset{258}{\overset{32;436^4}{32}}-32;258+\underset{258}{\overset{116;340^3}{116}}-116;\ 266+\underset{258}{\overset{22;412^2}{22}}-22;$ $258+\left\lceil\frac{258-16}{126-8}\right\rceil_{266}^{24;378^1}8-16\ \ R_7=266\ \ T1$
8	0	284	$R_8 = 292\ T2$
9	0	356	$356+\underset{356}{\overset{26;869^8}{26}}-26;356+\underset{356}{\overset{18;817^7}{18}}-18;356+\underset{356}{\overset{53;742^6}{53}}-53;356+\underset{356}{\overset{1;532^5}{1}}-1;356+\underset{356}{\overset{32;436^4}{32}}-32$ $356+\left\lceil\frac{356-116}{340-116}\right\rceil_{472}^{232;640^3}116-116;472+\left\lceil\frac{472-22}{206-11}\right\rceil_{483}^{33;618^2}11-22;483+\left\lceil\frac{483-24}{126-8}\right\rceil_{491}^{32;504^1}8-24$
	1	491	$491+\underset{491}{\overset{26;869^8}{26}}-26;491+\underset{491}{\overset{18;817^7}{18}}-18;491+\underset{491}{\overset{53;742^6}{18}}-18;491+\underset{491}{\overset{1;532^5}{1}}-1;491+\left\lceil\frac{491-32}{436-32}\right\rceil_{523}^{64;872^4}32-32$ $523+\underset{523}{\overset{232;640^3}{232}}-232;523+\underset{523}{\overset{33;824^2}{33}}-33;523+\left\lceil\frac{523-32}{126-8}\right\rceil_{531}^{40;630^1}8-32\ \ \ R_9=531\ \ T1$
10	0	549	$549+\underset{549}{\overset{26;869^8}{26}}-26;549+\underset{549}{\overset{26;869^8}{26}}-26;549+\underset{549}{\overset{18;817^7}{18}}-18;549+\underset{549}{\overset{53;742^6}{18}}-18;549+\left\lceil\frac{549-1}{532-1}\right\rceil_{550}^{2;1064^5}1-1;$ $550+\underset{550}{\overset{64;872^4}{64}}-64;550+\underset{550}{\overset{232;640^3}{232}}-232;550+\underset{550}{\overset{33;824^2}{33}}-33;550+\underset{550}{\overset{40;630^1}{40}}-40\ R_{10}=550\ \ T1$

El *STR* presentado de ejemplo anteriormente, es presentado en la *Tabla* 5. Se puede observar como el método RTA5, calcula 8 invariantes y 5 iteraciones, una menos en cada caso frente a RTA4 (*Tabla* 4) y un invariante menos y dos iteraciones menos que

RTA3 (*Tabla* 3). Además, se realizaron tests para RTA y RTA2, resultando 91 invariantes y 16 iteraciones para ambos casos.

4 Resultados Experimentales.

A continuación se presenta el desempeño del algoritmo RTA5, evaluado en conjunto con los métodos RTA/2/3/4 [8, 15, 16, 17]. Se emplearon dos métricas. La primera consiste en el número promedio de invariantes calculados para determinar la planificabilidad de un *STR*. La segunda, el número de iteraciones realizadas.

Tabla 5. Ejemplo RTA5 con orden por factor de utilización.

τ	q	t^q	$f(t^q)$; Orden de las tareas por FU_i (3 4 9 6 1 2 7 8 10 5)
1	0	0	$C_1 < D_1 \Rightarrow R_1 = 8$
2	0	8	$R_2 = 19$ *T2*
3	0	135	$135+\left\lceil\frac{135-8}{126-8}\right\rceil_{143}^{16;252^1}8-8;\ 143+{}_{143}^{11;206^2}11-11 \quad R_3=143$ *T1*
4	0	175	$R_4 = 175$ *T2*
5	0	176	$R_5 = 176$ *T2*
6	0	229	$229+{}_{229}^{116;340^3}116-116;229+{}_{229}^{32;436^4}32-32;229+{}_{229}^{16;252^1}16-16;229+\left\lceil\frac{229-11}{206-11}\right\rceil_{240}^{22;412^2}11-11;240+{}_{240}^{1;532^5}1-1$ $R_6=240$ *T1*
7	0	258	$258+{}_{258}^{116;340^3}116-116;258+{}_{258}^{32;436^4}32-32;258+{}_{258}^{53;742^6}18-18;258+\left\lceil\frac{258-16}{126-8}\right\rceil_{266}^{24;378^1}8-16$ $266+{}_{266}^{22;412^2}22-22;266+{}_{266}^{1;532^5}1-1 \quad R_7=266$ *T1*
8	0	284	$R_8 = 292$ *por T2*
9	0	356	$356+\left\lceil\frac{356-116}{340-116}\right\rceil_{472}^{232;640^3}116-116;472+\left\lceil\frac{472-32}{436-32}\right\rceil_{504}^{64;872^4}32-32;504+{}_{504}^{53;742^6}53-53;504+\left\lceil\frac{504-24}{126-8}\right\rceil_{520}^{40;630^1}8-24$ $520+\left\lceil\frac{520-22}{206-11}\right\rceil_{531}^{33;618^2}33-24;531+{}_{531}^{18;817^7}18-18;531+{}_{531}^{26;869^8}26-26;531+{}_{531}^{1;532^5}1-1 \quad R_9=531$ *T1*
10	0	549	$549+{}_{549}^{26;869^8}26-26;549+{}_{549}^{26;869^8}26-26;549+{}_{549}^{18;817^7}18-18;549+{}_{549}^{53;742^6}18-18;549+\left\lceil\frac{549-1}{532-1}\right\rceil_{550}^{2;1064^5}1-1;$ $550+{}_{550}^{64;872^4}64-64;550+{}_{550}^{232;640^3}232-232;550+{}_{550}^{33;824^2}33-33;550+{}_{550}^{40;630^1}40-40 \quad R_{10}=550$ *T1*

Se utilizaron grupos de *STR* de 10, 20 y 50 tareas, con distribuciones uniformes (*DU*) y exponenciales en grupos (*DEG*), y *FU* de {70, 75, 80, 82, 84, 85, 86, 88, 90, 92, 94, 96, 98} %. Para la *DU* se crearon tres grupos de *STR*, con periodos para las tareas entre 25 y 1k; 25 y 10k; 25 y 100k. Para la *DEG* se generaron 3 grupos, cuyos

periodos parten de 25 a 1k más 1k a 10k, para R2, se agregan períodos de 10k a 100k para R3 y finalmente se agregan períodos 100k a 1M para R4. Se simularon 10k de *STR* por cada *FU*.

4.1 RTA5 - Código Utilizado.

En la *Tabla* 6, se presenta el código en lenguaje Python, utilizado en la simulación.

Tabla 6. Código utilizado en lenguaje Python.

```
def rta5(rts):
    t_mas = rts[0]["R"] = rts[0]["C"]
    for task in rts[1:]:
        t_mas += task["C"]
        while t_mas > min([t["I"] for t in rts]):
            uf_sorted = sorted(rts[:rts.index(task)], key=lambda t: t["u"], reverse=True)
            for utask in uf_sorted:
                if t_mas > utask["I"]:
                    dif_a = t_mas - utask["A"]
                    a_mas_1 = ceil(dif_a / (utask["T"] - utask["C"]))
                    t_mas = (a_mas_1 * utask["C"]) + dif_a
                    utask["A"], utask["I"] = a_mas_1 * utask["C"], a_mas_1 * utask["T"]
                    if t_mas > task["D"] return False
        task["R"] = t_mas
    return True
```

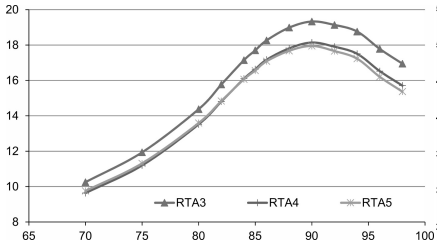

Fig. 1. 10 Tareas y períodos 25-10K *DU*

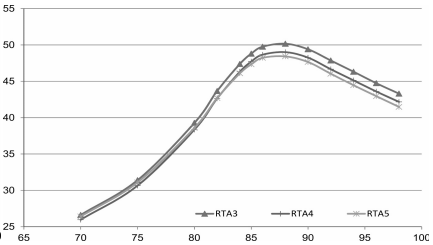

Fig. 2. 20 Tareas y períodos 25-10K *DU*

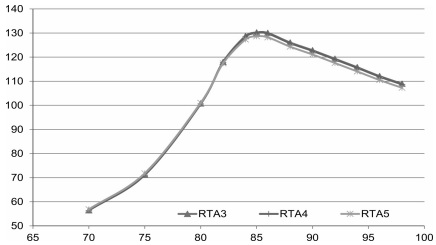

Fig. 3. 50 Tareas y períodos 25-10K *DU*

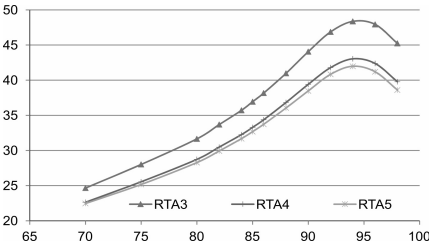

Fig. 4. 10 Tareas R2 *DEG*

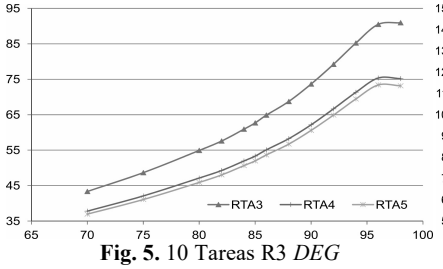

Fig. 5. 10 Tareas R3 *DEG*

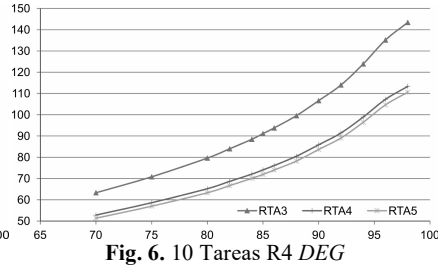

Fig. 6. 10 Tareas R4 *DEG*

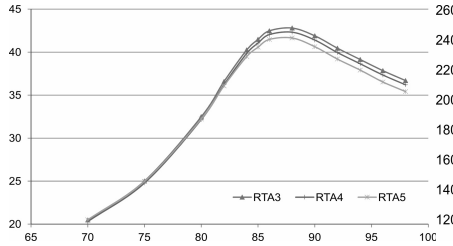

Fig. 7. 20 Tareas R2 *DEG*

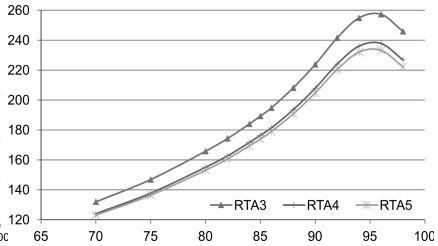

Fig. 8. 20 Tareas R3 *DEG*

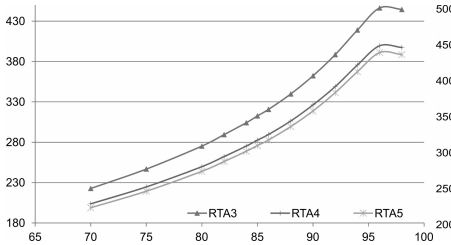

Fig. 9. 20 Tareas R4 *DEG*

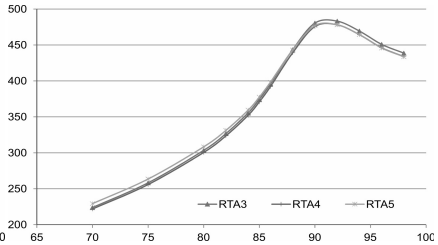

Fig. 10. 50 Tareas R2 *DEG*

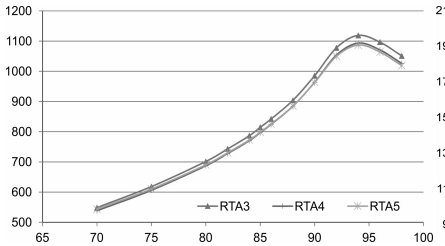

Fig. 11. 50 Tareas R3 *DEG*

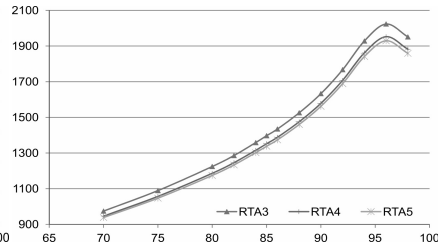

Fig. 12. 50 Tareas R4 *DEG*

5 Conclusiones

RTA5 mejora en promedio los métodos simulados, como se puede observar en las Figuras 1 a la 12. No obstante, no en todos los casos puede sacar ventaja, como se puede observar en la *Tabla* 7. En conjuntos con tareas con alto factor de utilización mejora su eficiencia y el número de iteraciones es menor para sistemas no-planificables sobre los planificables, lo cual contribuye a que sea una mejor opción para aplicaciones en línea.

Sólo se presentaron las gráficas más relevantes y en estas se puede apreciar una mejora a RTA4 y a RTA3. Se debe tener en cuenta que RTA y RTA2 son métodos de orden n^2 en su complejidad. Sin embargo, se puede apreciar como RTA3/4/5 reducen el *CC* considerablemente. No es posible presentar todas las extensas simulaciones y análisis realizados, por cuestiones de espacio.

Como trabajos futuros, se evaluará la implementación de RTA5 en diversas plataformas de desarrollo de sistemas embebidos, analizando el costo temporal del método en la práctica.

Tabla 7. Resultados con *FU*=86% en Invariantes (*CC*) y Número de iteraciones (*N*).

		R1-1K		*R1-10K*		*R1-100K*		*R2*		*R3*		*R4*	
	Método	*CC*	*N*	*CC*	*N*	*CC*	*N*	*CC*	*N.*	*CC*	*N*	*CC*	*N*
	RTA	102,3	17,2	111,1	18,7	112,5	19,0	161,8	25,1	223,6	33,6	287,1	42,4
	RTA2	85,9	15,2	94,8	16,7	96,1	17,0	137,1	21,9	185,0	28,7	233,2	35,5
10	RTA3	16,0	7,0	18,3	8,1	18,7	8,4	38,2	11,5	64,9	17,1	93,8	22,9
	RTA4	15,4	6,7	17,2	7,4	17,4	7,5	34,4	10,1	55,1	13,7	76,1	17,3
	RTA5	15,3	6,0	17,1	6,9	17,3	7,0	33,7	9,9	53,6	14,0	74,0	18,0
	RTA	425,8	35,3	467,1	39,0	474,9	39,8	425,8	35,3	923,7	68,4	1239,0	89,0
	RTA2	349,9	30,9	391,1	34,6	398,5	35,3	349,9	30,9	746,4	57,2	963,6	71,8
20	RTA3	42,5	13,9	49,8	16,4	50,9	17,0	42,5	13,9	194,7	32,4	320,7	44,6
	RTA4	42,0	13,7	48,7	15,8	49,5	16,1	42,0	13,7	181,4	29,3	289,6	39,0
	RTA5	41,5	12,3	48,2	14,9	49,1	15,3	41,5	12,3	178,9	30,0	282,8	40,5
	RTA	2648,9	88,5	2906,8	98,3	2988,2	101,5	4372,1	131,3	5970,8	173,7	7756,7	220,1
	RTA2	2156,0	76,7	2438,8	87,1	2525,0	90,5	3526,0	110,4	4680,0	141,8	5880,4	173,9
50	RTA3	130,0	33,8	164,7	40,3	171,8	42,4	395,9	56,5	842,1	79,8	1434,7	105,2
	RTA4	129,7	33,7	163,6	39,8	170,2	41,4	392,1	55,9	824,1	77,5	1388,3	100,5
	RTA5	128,2	30,5	161,7	38,3	168,2	40,2	398,2	55,3	825,3	80,1	1373,8	105,9

Referencias

[1] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* vol. 20, pp. 46-61, 1973.

[2] E. Bini, G. Buttazzo, and G. Buttazzo, "A Hyperbolic Bound for the Rate Monotonic Algorithm," *IEEE Transactions on Computer,* vol. 52, pp. 933-942, 2003.

[3] J. A. Stankovic, "Misconceptions About Real-Time Computing: A Serius Problem for Next-Generations Systems," *IEEE Computer,* vol. Octubre, pp. 10-19, 1988.

[4] J. M. Urriza, R. Cayssials, and J. D. Orozco, "Modelado de Sistemas de Tiempo Real Planificados por RM o DM: Caracterización y Análisis," in *XXXIV Conferencia Latinoamericana de Informática, CLEI 2008*, Santa Fe, Argentina, 2008, pp. 1435-1444.

[5] M. Joseph and P. Pandya, "Finding Response Times in Real-Time System," *The Computer Journal (British Computer Society),* vol. 29, pp. 390-395, 1986.

[6] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "Applying New Scheduling Theory to Static Priority Preemptive Scheduling," *Software Engineering Journal,* vol. 8, pp. 284-292, 1993.

[7] J. Santos and J. D. Orozco, "Rate Monotonic Scheduling in Hard Real-Time Systems," *Information Processing Letters,* vol. 48, pp. 39-45, 1993.

[8] M. Sjödin and H. Hansson, "Improved Response-Time Analysis Calculations," in *IEEE 19th Real-Time Systems Symp.*, 1998, pp. 399-409.

[9] R. I. Davis, "A review of fixed priority and EDF scheduling for hard real-time uniprocessor systems," *SIGBED Rev.,* vol. 11, pp. 8–19, 2014.

[10] R. I. Davis, A. Zabos, and A. Burns, "Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems," *IEEE Transactions on Computers,* vol. 57, pp. 1261-1276, 2008.

[11] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Department of Statistics, Carnegie-Mellon, Pitsburg, USA, Internal Report1987.

[12] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," in *IEEE Real-Time Systems Symposium*, 1989, pp. 166-171.

[13] L. Wan-Chen, H. Jen-Wei, S. Wei-Kuan, and K. Tei-Wei, "A faster exact schedulability analysis for fixed-priority scheduling," *Journal of Systems and Software,* vol. 79, pp. 1744-1753, 2006.

[14] J. M. Urriza, C. Ricardo, and J. D. Orozco, "Test Rápido de Planificabilidad para R.M. o D.M.," in *XXXI Conferencia Latinoamericana de Informática. CLEI*, Colombia, Cali, 2005.

[15] J. M. Urriza, F. E. Paez, J. D. Orozco, and R. Casyssials, "Computational Cost Reduction for Real-Time Schedulability Tests Algorithms," *IEEE Latin America Transactions,* vol. 13, pp. 3714-3723, 2015.

[16] J. M. Urriza, F. E. Páez, M. Ferrari, R. Cayssials, and J. D. Orozco, "A New RM/DM Low Cost Schedulability Test," in *Eight Argentine Symposium and Conference on Embedded Systems*, Buenos Aires, 2017, pp. 13-18.

[17] J. M. Urriza, J. D. Orozco, R. Cayssials, and L. Schorb, "Reduced Computational Cost in the Calculation of Worst Case Response Time for Real Time Systems," *Journal of Computer Science & Technology,* vol. 9, pp. 72-81, 2009.