UNIVERSIDAD DE OVIEDO

Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

Tesis doctoral

TÉCNICAS ESTOCÁSTICAS PARA EL CÁLCULO DEL TIEMPO DE RESPUESTA EN SISTEMAS DE TIEMPO-REAL

presentada por

Jose Luis Díaz de Arriba

para la obtención del grado de

Doctor por la Universidad de Oviedo

Director: Daniel Fernando García Martínez

Gijón, Mayo 2003

A mi mujer; y a mi hijo
que verá la luz a la vez que esta Tesis.

Índice general

Índice de figuras

Índice de cuadros

Notación

Notación para el modelo y análisis

A Matriz que se deriva de **P**, que resume la relación de recurrencia existente entre las componentes de $\boldsymbol{\pi}$. (Ver pág. 109)

$b_i(j)$ Elemento de la matriz de Markov **P**, situado en la columna i, fila j. (Ver pág. 94)

$\mathcal{B}_i$ Bloqueo que puede sufrir la tarea τ_i como consecuencia del acceso a recursos compartidos. Es una variable aleatoria. (Ver pág. 144)

$\mathcal{C}$ Tiempo de ejecución (variable aleatoria), igual al tiempo que requeriría la tarea si no hubiese otras en el sistema. $\mathcal{C}_i$ es el tiempo de ejecución de la tarea i-ésima, en un sistema de tareas periódicas. $\mathcal{C}_j$ es el tiempo de ejecución del trabajo j-ésimo que se activa en el sistema desde el origen de tiempos. (Ver pág. 42)

$C^{\mathbf{max}}$ Tiempo de ejecución máximo de una tarea (*WCET*). (Ver pág. 43)

$\bar{C}$ Tiempo de ejecución promedio de una tarea. (Ver pág. 57)

$C^{\mathbf{min}}$ Tiempo de ejecución mínimo de una tarea. (Ver pág. 43)

D_i Plazo (o *deadline*) relativo de la tarea i-ésima en un sistema de tareas periódicas. (Ver pág. 42)

$\mathcal{D}_{i,k}$ Duración de la sección crítica guardada por el semáforo S_k en la tarea τ_i. Es una variable aleatoria. (Ver pág. 146)

D[*a*] Distribución determinista. Representa una variable aleatoria que sólo puede tomar el valor a, con probabilidad 1.

Φ_i Fase (u *offset*) de la tarea i-ésima en un sistema de tareas periódicas. (Ver pág. 42)

$f_{\mathcal{X}}(\cdot)$ Función de probabilidad (PF) de la variable aleatoria $\mathcal{X}$, también llamada "función de masa de probabilidad" (PMF). (Ver pág. 43)

$F_{\mathcal{X}}(\cdot)$ Función de distribución acumulada (CDF) de la variable aleatoria $\mathcal{X}$. (Ver pág. 44)

$f^{[a..b]}(\cdot)$ Función truncada a un intervalo. Toma el valor de f dentro del intervalo $[a..b]$ y cero fuera de él. (Ver pág. 70)

Γ Trabajo. Instancia de una tarea. A veces lleva dos subíndices, como en $\Gamma_{i,j}$, para indicar que se trata de la j-ésima activación de la tarea i-ésima en un modelo de tareas periódicas. Otras veces lleva un único subíndice, como en Γ_j, cuando no es importante de qué tarea proviene, para indicar que es el j-ésimo trabajo que se activa desde el origen de tiempos. (Ver pág. 45)

i Índice generalmente usado para referirse a los parámetros de una tarea, como en $\mathcal{C}_i$, D_i, etc. (Ver pág. 45)

j Índice generalmente usado para referirse a los parámetros del trabajo que hace el lugar j-ésimo dentro de la secuencia de trabajos, cuando no importa la tarea de la cual proviene. (Ver pág. 45)

$\mathcal{J}_i$ *Jitter* de la tarea τ_i. Es una variable aleatoria que se suma al instante teórico de activación para generar el instante real de activación. (Ver pág. 159)

J_i^{max}, J_i^{min} Valores máximo y mínimo que puede tomar el *jitter* de la tarea τ_i. (Ver pág. 159)

$\lambda_{i,j}$ Instante de activación de la instancia j-ésima de la tarea τ_i. (Ver pág. 45)

λ_j Instante de activación del trabajo j-ésimo desde el origen de tiempos. (Ver pág. 45)

λi Valores propios de la matriz $\mathbf{A}$. (Ver pág. 109)

m_r Índice de fila del último elemento no nulo de la columna r-ésima de la matriz de Markov $\mathbf{P}$. (Ver pág. 94)

N Número total de tareas en un sistema de tareas periódicas. (Ver pág. 42)

$\mathcal{O}$ Variable aleatoria "nula". Sólo puede tomar el valor cero con probabilidad 1. (Ver pág. 134)

$\mathbf{P}$ Matriz de Markov que gobierna el proceso de la carga $\{\mathcal{W}(k)\}$. (Ver pág. 94)

$\boldsymbol{\pi}$ Vector columna solución de la ecuación $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$. Si sus componentes suman 1, es la función de probabilidad de la carga existente al inicio del hiperperiodo, en régimen estacionario. (Ver pág. 96)

P_i Prioridad de la tarea τ_i, en políticas de asignación de prioridades a nivel de tareas, como RM o DM. (Ver pág. 41)

P_j Prioridad del trabajo que llega en j-ésimo lugar desde el origen de tiempos, en políticas de asignación de prioridades a nivel de trabajo, como EDF. (Ver pág. 41)

PCP Probabilidad de que una tarea cumpla su plazo. (Ver pág. 46)

PPP Probabilidad de que una tarea pierda su plazo. (Ver pág. 46)

$\mathbb{P}\{x\}$ Probabilidad del suceso x. (Ver pág. 43)

$\mathcal{R}_i$ Tiempo de respuesta de la tarea τ_i. Es una variable aleatoria. (Ver pág. 45)

$\mathcal{R}_j$ Tiempo de respuesta del trabajo Γ_j, es una variable aleatoria. (Ver pág. 45)

$\mathcal{R}_j^{\langle k \rangle}$ Tiempo de respuesta del trabajo Γ_j obtenido en la k-ésima iteración del método "partir, convolucionar y juntar", igual al tiempo de respuesta que presentaría dicho trabajo si sólo pudieran expulsarle los k siguientes. Es una variable aleatoria. (Ver pág. 76)

r Máximo tiempo libre. Tiempo durante el cual el sistema está ocioso en el primer hiperperiodo completo (definición 9), en el caso de que todas las tareas soliciten su tiempo de ejecución mínimo. La matriz de Markov $\mathbf{P}$ tiene una estructura regular a partir de su columna r-ésima. (Ver pág. 95)

S Sistema, conjunto de tareas periódicas. (Ver pág. 42)

S_i Semáforo que guarda el acceso a un cierto recurso en un sistema en el que las tareas pueden compartir recursos. (Ver pág. 146)

s Mínimo tiempo libre. Tiempo durante el cual el sistema está ocioso en el primer hiperperiodo completo (definición 9), en el caso de que todas las tareas soliciten su tiempo de ejecución máximo. (Ver pág. 90)

T Longitud del hiperperiodo, igual al mínimo común múltiplo de los periodos de las tareas, en un sistema de tareas periódicas. (Ver pág. 54)

τ_i Tarea i-ésima en un sistema de tareas periódicas. (Ver pág. 42)

T_i Periodo de la tarea i-ésima en un sistema de tareas periódicas. (Ver pág. 42)

$U^{\mathbf{min}}$ Factor de utilización mínima del sistema. (Ver pág. 57)

$\bar{U}$ Factor de utilización promedio del sistema. (Ver pág. 57)

$U^{\mathbf{max}}$ Factor de utilización máxima del sistema. Coincide con el factor de utilización usado en el análisis clásico de peor caso. (Ver pág. 57)

$\mathbf{U}[a,b]$ Distribución uniforme discreta. Representa una variable aleatoria que sólo puede tomar los valores enteros $a, a+1, \ldots, b-1, b$, todos ellos con la misma probabilidad.

$\mathbf{v}_i$ Vectores propios de la matriz $\mathbf{A}$. (Ver pág. 109)

$\mathcal{W}(t)$ Carga existente en el sistema en el instante t. Es una variable aleatoria. (Ver pág. 52)

$\mathcal{W}_k$ Carga observada al inicio del hiperperiodo k-ésimo, es decir, en el instante kT. Es una variable aleatoria. La secuencia $\{\mathcal{W}_k\}$ es un proceso estocástico denominado "proceso de la carga". (Ver pág. 89)

$W^{\mathbf{max}}$ Carga observada al final del primer hiperperiodo completo (definición 9), si todas las tareas activadas en él requiriesen su tiempo de ejecución máximo. (Ver pág. 90)

$W^{\mathbf{min}}$ Carga observada al final del primer hiperperiodo completo (definición 9), si todas las tareas activadas en él requiriesen su tiempo de ejecución mínimo. (Ver pág. 95)

Notación para el análisis de complejidad

b Número de puntos que definen la función de probabilidad de la carga al inicio del hiperperiodo estacionario. (Ver pág. 181)

k Número promedio de expulsiones que un trabajo cualquiera puede sufrir antes de la expiración de su plazo. (Ver pág. 185)

k_j Número promedio de expulsiones que el trabajo j-ésimo puede sufrir antes de la expiración de su plazo. (Ver pág. 185)

m Número máximo de puntos que definen la función de probabilidad del tiempo de ejecución de las tareas de un sistema. (Ver pág. 180)

n Número total de trabajos activados en un hiperperiodo completo. (Ver pág. 180)

$\bar{T}$ Tiempo promedio entre activaciones de trabajos. (Ver pág. 180)

Operadores y relaciones

$\otimes$ Operador de convolución discreta. (Ver pág. 235)

$\succcurlyeq$ Relación “peor que”, que permite comparar el pesimismo relativo de dos variables aleatorias, en el contexto de tiempo real. (Ver pág. 132)

1. Introducción

You may delay, but time will not

(Benjamin Franklin)

Se presenta en este capítulo una breve introducción a los sistemas de tiempo real (sección 1.1) con especial hincapié en la problemática de la planificación o *scheduling* (sección 1.2), para finalmente, en la sección 1.3, centrar el problema del que se ocupa la presente tesis.

1.1. Sistemas de tiempo real

La definición clásica de un sistema de tiempo real dice que es aquel en el que, para que su respuesta se considere correcta, no sólo se tienen en cuenta los resultados lógicos de las computaciones, sino también el instante en que estos resultados son producidos [Stankovic, 1988].

Por tanto, cada tarea desempeñada por el sistema no sólo tiene una especificación funcional, sino también una especificación temporal, que a menudo consiste tan solo en su plazo (en inglés *deadline*), es decir, el tiempo dentro del cual la tarea debe finalizar, medido desde el instante de la activación de la tarea. Las consecuencias que puede acarrear la violación del plazo son variadas, según el sistema con que estemos tratando. En muchos casos se supone que una vez transcurrido el plazo, ya no tiene sentido continuar la ejecución de esa tarea porque el resultado deja de ser útil. En otros casos la violación del plazo puede traer consecuencias más graves ya que los sistemas de tiempo real interactúan con su entorno y la incapacidad de reaccionar en el momento adecuado puede causar la destrucción del sistema y la pérdida de vidas humanas.

Según la gravedad que revista la violación de los plazos éstos suelen clasificarse [Stankovic y Ramamritham, 1990; Audsley y Burns, 1990][1] en:

[1] No todos los autores adoptan esta clasificación. Para la mayoría, un plazo duro es lo mismo que uno crítico, es decir, se considera plazo duro al que debe cumplirse para evitar consecuencias catastróficas. Para aumentar más la confusión en la nomenclatura, los mismos autores que clasifican los plazos según el esquema anterior, usan el término "Sistema de tiempo real duro" (*"hard real–time system"*) para los sistemas cuyo fallo puede causar catástrofe. Usando su propia terminología serían sistemas que contienen *plazos críticos*.

- Blandos (*soft deadline*). Aún cuando finalice después de su plazo, el resultado puede ser útil, si bien la máxima utilidad se obtiene dentro del plazo.
- Duros (*hard deadline*). El resultado no tiene utilidad si no se obtiene antes del plazo.
- Críticos (*critical deadline*). Si no se obtiene un resultado dentro del plazo las consecuencias son catastróficas.

El estudio teórico de los sistemas de tiempo real trata de determinar mediante un análisis a priori si se cumplirán todos los plazos duros y críticos. Teniendo en cuenta que muchas tareas se ejecutarán sobre un mismo procesador, y que por tanto interferirán en el tiempo unas con otras, la demostración de que se cumplirán todos los plazos bajo cualquier circunstancia no es en absoluto trivial.

La investigación en el campo del tiempo real se ha centrado en los siguientes aspectos [Shin y Ramanathan, 1994], que en muchas ocasiones se solapan entre sí:

- *Scheduling* (planificación): Conociendo de antemano todas las tareas que se van a ejecutar, sus plazos, relaciones de precedencia, etc. así como el uso de recursos que necesita cada una de ellas, el problema del *scheduling* es el encontrar para cada tarea un recurso que la ejecute y un instante en el tiempo en el que ha de comenzar su ejecución, de modo que se garantice el cumplimento de todos los plazos[2]. Como se ha dicho, el problema es muy difícil (en su forma general es NP–completo) por lo que los primeros estudios restringieron fuertemente el tipo de sistemas a estudiar, y en los posteriores trabajos se intentaron relajar estas restricciones.

  El *scheduling* en un solo procesador consiste en asignar solamente el instante de arranque de cada tarea (o bien idear un algoritmo que decida esto en tiempo de ejecución del sistema) y demostrar que usando esa asignación o ese algoritmo se cumplirán todos los plazos. Incluso este problema está sin solucionar para el caso general.

  En muchas aplicaciones reales las tareas no se ejecutan una sola vez, sino que se "activan" repetidas veces en diferentes instantes, siguiendo algún tipo de patrón. El caso más sencillo es el de las tareas periódicas, para las cuales la distancia (temporal) entre dos sucesivas activaciones es constante. En este caso el problema de la planificación consiste en decidir cuál de las tareas "activadas" recibe el recurso, y cuáles deben dejarse a la espera, de modo que se garantice también el cumplimiento de los plazos. O inversamente, dado

[2] A diferencia del *scheduling* clásico en el que se busca minimizar el tiempo de ejecución. Puede mostrarse fácilmente [Shin y Ramanathan, 1994] que ambos criterios no son equivalentes, sino que pueden ser contrapuestos

un algoritmo de planificación, diseñar un análisis que nos permita decidir si todas las tareas cumplirán sus plazos o no.

- Arquitecturas de tiempo real: Para que el análisis a priori sea posible, es necesario que todos los tiempos de ejecución estén acotados y que sean lo más deterministas posible, o dicho de otro modo, que su comportamiento sea predecible. Esto hace que resulten inadecuados la mayoría de los procesadores de propósito general actuales, debido a la existencia de jerarquías de memoria, cauces de ejecución, etc. que hacen bastante impredecible el tiempo de ejecución incluso de las instrucciones máquina[3].

  Por esta razón se ha investigado en el diseño de arquitecturas más adecuadas para el tiempo real, en el sentido de "más predecibles". Sin embargo, la tendencia actual es utilizar procesadores de propósito general y tratar de tener en cuenta los efectos de las caches, *pipelines*, etc. en el análisis de planificabilidad. Esto suele llevar a análisis excesivamente pesimistas, por lo que también se investiga en cómo reducir este pesimismo sin comprometer la aplicabilidad del análisis.

- Comunicaciones de tiempo real: De nuevo se trata de idear protocolos que proporcionen tiempos de respuesta acotados, para permitir el análisis.

- Sistemas operativos de tiempo real: los sistemas operativos deben ser diseñados con un nuevo paradigma en mente [Stankovic, 1996], que ha de ser el mantener un equilibrio entre flexibilidad y predictibilidad. Las primitivas del operativo deben tener un tiempo de respuesta perfectamente acotado, y a la vez ser lo suficientemente flexibles como para que el operativo sea de propósito general, y no un micro núcleo a la medida de la aplicación.

- Tolerancia a fallos. Ya que a menudo los sistemas de tiempo real son críticos en el sentido antes expuesto, debe tenerse en cuenta en su diseño que el *hardware* sobre el que se ejecutan puede fallar. Para contemplar tal eventualidad la estrategia típica es la replicación de elementos, de modo que si uno falla, una de sus copias pueda hacerse cargo de la situación. Evidentemente son necesarios algoritmos para detectar los fallos y tomar las medidas oportunas dentro del plazo correspondiente, por lo cual la tolerancia a fallos debe tenerse en cuenta también en el análisis.

Nos centraremos a partir de ahora exclusivamente en el problema de la planificación o *scheduling* y su análisis asociado.

[3] Para un análisis detallado y una comparativa de la idoneidad de las CPUs de propósito general para tiempo real, véase [Weems y Dropsho, 1995].

1.2. Planificación en los sistemas de tiempo real

El problema de la planificación debe tener en cuenta varios aspectos diferentes:

- Recursos. ¿Cuál es el recurso por el que los consumidores compiten? En general suelen considerarse los recursos *hardware*, como el procesador, la memoria, el almacenamiento masivo (disco), el sistema de comunicaciones (red).

  La arquitectura del sistema (monoprocesador, multiprocesador de memoria compartida, o de memoria distribuida, sistema distribuido homogéneo o heterogéneo,...) también es un parámetro importante.

- Consumidores. ¿Quienes compiten por esos recursos? Se trata en general de las tareas que deben realizarse en el sistema, las cuales requieren un cierto tiempo de procesador, espacio de memoria, etc. Estas tareas pueden tener características especiales, en cuanto a los instantes en que "llegan" al sistema, y así pueden ser periódicas, aperiódicas, mantener relaciones de precedencia entre sí, etc. Por otro lado pueden ser independientes o bien compartir un recurso (aparte del procesador), en cuyo caso deben coordinarse para no acceder a él a la vez.

  Por otro lado sus restricciones de tiempo real también pueden ser muy diversas. Pueden tener asociado un plazo estricto, duro o blando, y este plazo puede coincidir con el periodo de la tarea (en caso de tareas periódicas), o ser menor, o incluso ser mayor lo que significaría que una nueva instancia de la tarea puede llegar mientras la anterior aún está en ejecución. Todos estos son también parámetros del problema de la planificación.

- Algoritmos. ¿Cómo decidir qué tarea debe recibir el recurso y en qué momento quitárselo para dárselo a otra tarea? Existen diferentes algoritmos de planificación, que iremos viendo en detalle. A veces el problema de la planificación consiste precisamente en encontrar un nuevo algoritmo para este cometido, pero otras muchas veces se trabaja con un algoritmo prefijado y el problema de la planificación se reduce entonces al análisis, que es el siguiente aspecto.

- Análisis. Dados un conjunto de recursos, un conjunto de tareas y un algoritmo de planificación el análisis consiste en determinar matemáticamente si las tareas van a cumplir siempre sus plazos, o si por el contrario existe la posibilidad de que alguna de ellas pierda su plazo. A veces hallar la respuesta a tal pregunta no es posible, si no se imponen algunas restricciones al sistema y al algoritmo de planificación. En ocasiones, con las restricciones adecuadas, la respuesta puede hallarse mediante un sencillo cálculo basado en la

utilización del sistema (el porcentaje de tiempo en el cual el sistema estará ocupado). En otras ocasiones los cálculos pueden ser más complejos.

En la mayoría de los análisis propuestos en la literatura, la respuesta que se obtiene es de tipo "si/no", es decir, o bien el sistema es planificable (lo que significa que bajo cualquier circunstancia se garantiza que todas las tareas cumplirán sus plazos), o bien es no planificable (lo que significa que existe una posibilidad de que alguna de ellas pierda su plazo). En esta tesis se propone un nuevo enfoque estadístico, por el cual lo que se obtiene es la probabilidad de que cada tarea cumpla su plazo. Si estas probabilidades son del 100 % para todas las tareas, el sistema sería planificable en el sentido clásico, pero incluso si alguna de las probabilidades es inferior al 100 % el sistema podría seguir siendo válido para aplicaciones no tan críticas.

En la figura 1.1 se muestran de forma esquemática todos estos aspectos involucrados en el problema de la planificación. En las secciones siguientes explicaremos con más detalle algunos de estos aspectos. Finalmente, en la sección 1.3 concretaremos cuáles de estos aspectos son los que se tendrán en cuenta en la presente tesis, y cuáles se dejan fuera.

1.2.1. Recursos y arquitectura

Los recursos son el *hardware* del sistema, como los procesadores, redes de comunicaciones, la memoria, los elementos de almacenamiento masivo, etc.

El recurso memoria no suele considerarse en los sistemas de tiempo real, pues se considera que la asignación se realiza de forma estática y puede ignorarse este recurso una vez el sistema empieza a funcionar. La asignación dinámica de memoria plantea problemas en los sistemas de tiempo real, puesto que suele basarse en mecanismos de paginación que incorporarían una gran sobrecarga en caso de fallo de página (y que debería tenerse en cuenta para el análisis del peor caso posible, que se haría por tanto mucho más pesimista).

El acceso a sistemas de almacenamiento masivo (discos) no suele ser crítico en muchos de los sistemas de tiempo real, ya que se produce raramente, con la excepción de los sistemas de tiempo real transaccionales o multimedia.

El recurso más ampliamente considerado en la teoría de la planificación es el procesador. No obstante, algunos autores [Tindell y Clark, 1994] han mostrado que el recurso de comunicaciones (red), puede considerarse bajo ciertas circunstancias como un procesador, en cuanto a la planificación se refiere, por lo que muchos de los resultados existentes sobre procesadores son también aplicables a las comunicaciones. Ambos tipos de recurso son unitarios en el sentido de que en un instante dado el recurso sólo puede estar asignado a una tarea a lo sumo.

Los recursos se organizan de acuerdo a una determinada arquitectura. Las más habituales son:

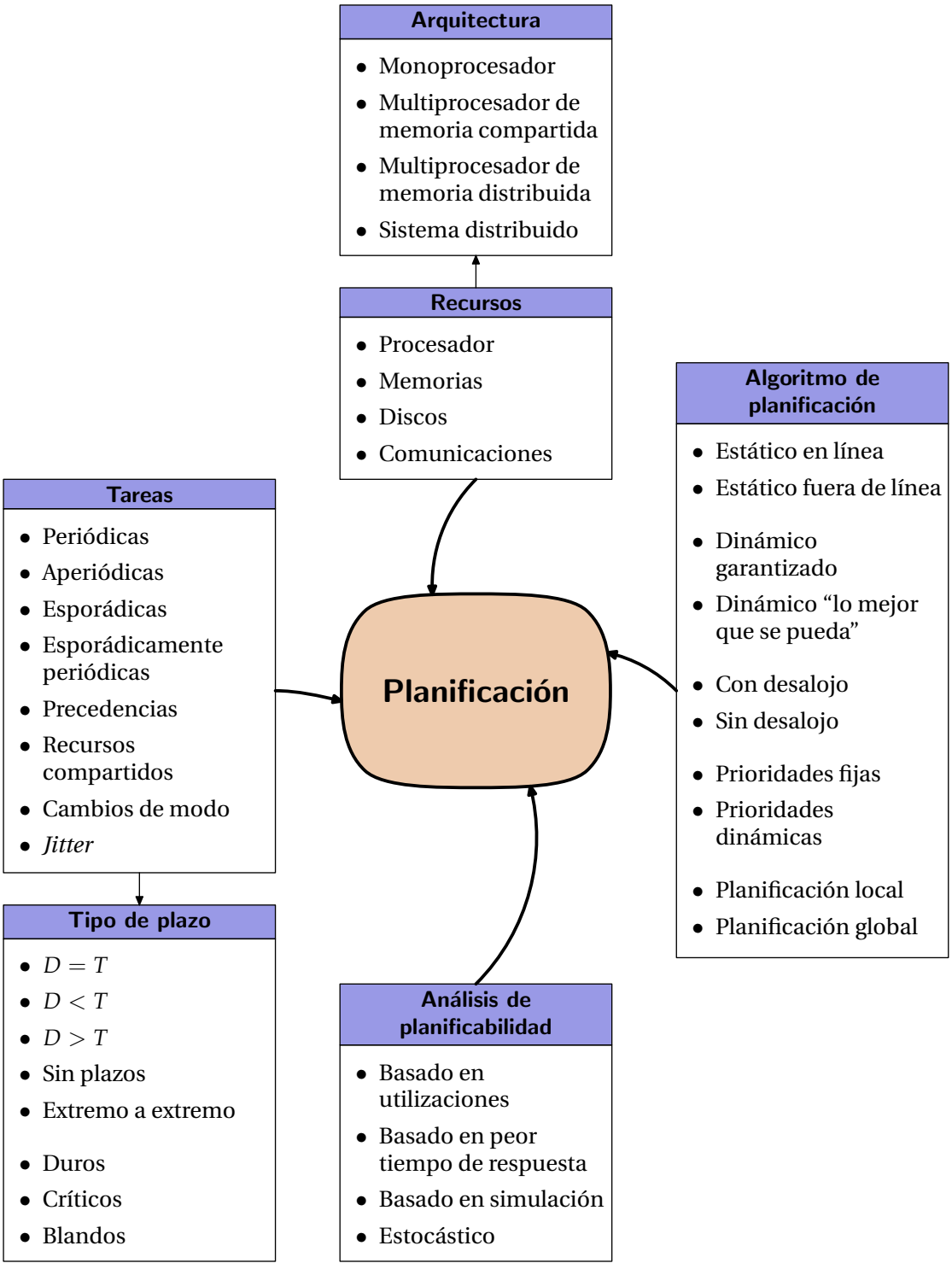


Figura 1.1.: Aspectos involucrados en el problema de la planificación en los sistemas de tiempo real

- **Monoprocesador**. El sistema está formado por un único procesador, y no hay red de comunicación.
- **Multiprocesador con memoria compartida**. El sistema tiene más de un procesador que comparten la memoria principal o parte de ella. Esta memoria compartida es utilizada como mecanismo de comunicación entre los procesadores.
- **Multiprocesador con memoria distribuida**. El sistema tiene más de un procesador, y cada uno de ellos con su memoria local. No existe memoria compartida. La comunicación entre procesadores ocurre a través de canales específicos.
- **Sistemas distribuidos**. Conceptualmente análogos a los sistemas multiprocesadores con memoria distribuida, se diferencian básicamente en que en lugar de procesadores con memoria local, se utilizan computadores autónomos completos. Esto puede ocasionar una gran heterogeneidad en los recursos, ya que cada computador puede tener diferente velocidad, memoria, juego de instrucciones, etc. La comunicación entre estos computadores se realiza a través de redes de comunicación que generalmente suelen ser más lentas que los canales dedicados en los sistemas multiprocesador con memoria distribuida.

1.2.2. Consumidores de recursos (tareas)

Una tarea es en realidad una abstracción del concepto de proceso o hilo típico de los sistemas operativos multitarea. En el contexto de los sistemas de tiempo real una tarea viene definida por una secuencia de peticiones de recursos. Mientras el recurso no sea concedido, la tarea se detiene, cuando el recurso es concedido (por el planificador), la tarea lo utiliza durante un tiempo (que el planificador decide), lo que le permite avanzar en su secuencia.

Para que una tarea reciba un recurso han de cumplirse dos condiciones: 1) la tarea ha solicitado el recurso; 2) la tarea no está pendiente de ninguna condición de sincronización. Pueden existir varias tareas que cumplan en un momento dado las dos condiciones anteriores. Es misión del planificador elegir una de ellas.

Las condiciones de sincronización a que se hace referencia en el párrafo anterior aparecen sólo si las tareas dependen unas de otras. Por ejemplo, si existen relaciones de precedencia, una tarea no podrá comenzar su ejecución hasta que otras tareas de las que depende no hayan terminado la suya. Otro caso es cuando existe acceso a recursos compartidos. En este caso, la zona de código de la tarea que necesita acceder al recurso se marca como *zona crítica*. Una tarea no podrá entrar a ejecutar su zona crítica, hasta que las restantes tareas no hayan salido

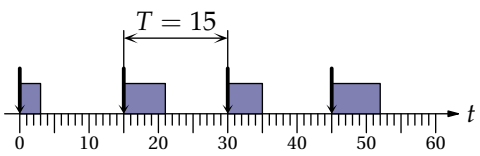
de sus respectivas zonas críticas relacionadas con ese mismo recurso. Las tareas que no tienen asociada ninguna condición de sincronización se denominan *independientes*. En caso contrario se habla de tareas dependientes. Evidentemente un sistema compuesto únicamente por tareas independientes es más sencillo de analizar que otro en el que existan tareas dependientes.

Hemos dicho que el recurso más ampliamente considerado en el problema de la planificación es el procesador. En este contexto, una tarea puede verse como una secuencia de "peticiones de procesador". Cada una de estas peticiones se denomina tradicionalmente "**trabajo**" En cada una de las peticiones, la tarea demanda una cantidad de tiempo de procesador que denotaremos por C y que se denomina "tiempo de ejecución"[4]. Este tiempo puede interpretarse como lo que tardaría en ejecutarse esa activación de la tarea si tuviera el procesador para ella sola. Cabe observar que C puede tomar un valor distinto en cada activación, si bien el análisis clásico considera C una cantidad constante igual al peor tiempo de ejecución posible, esto es, al valor más alto posible de C. En esta tesis propondremos un modelo en el que C es una variable aleatoria y por tanto puede tomar cualquier valor, siguiendo una distribución estadística arbitraria, pero conocida.

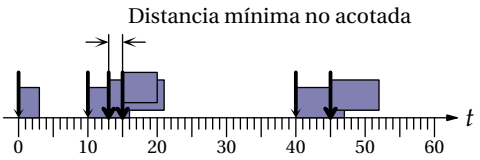
Los instantes en los cuales se solicita el procesador se denominan "instantes de activación" de la tarea (o trabajo), o también "instantes de llegada". Podemos clasificar las tareas en varios tipos atendiendo a la regularidad en los instantes de llegada (véase la Figura 1.2):

- **Tareas periódicas**. Cada T unidades de tiempo se solicita el procesador. El parámetro T se denomina *periodo* de la tarea. Este tipo de tareas suelen activarse por medio del reloj del sistema [Buttazzo, 1997].
- **Tareas aperiódicas**. La petición de procesador puede ocurrir en cualquier instante, desconocido a priori. Este tipo de tareas suelen activarse por eventos impredecibles, externos al sistema [Buttazzo, 1997]
- **Tareas esporádicas**. Las llegadas ocurren de forma no periódica, pero existe un tiempo mínimo entre dos llegadas consecutivas. La principal diferencia entre las tareas esporádicas y las aperiódicas es que sobre las primeras es posible encontrar un "peor caso" que es cuando llegan con el mínimo intervalo posible, y en este sentido tratarlas como periódicas. En cambio, para las aperiódicas, es imposible realizar ningún tipo de análisis de tiempo real.
- **Tareas esporádicamente periódicas**. Las peticiones ocurren por ráfagas. Dentro de cada ráfaga, las peticiones se hallan separadas por un intervalo fijo, como si fueran periódicas. Las ráfagas a su vez ocurren de forma esporádica, es decir, impredecible pero con una distancia mínima entre dos ráfagas

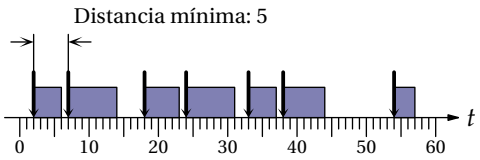
[4] En muchos textos también se usa el término "tiempo de computación", no confundir con el tiempo de respuesta del que se habla más adelante.



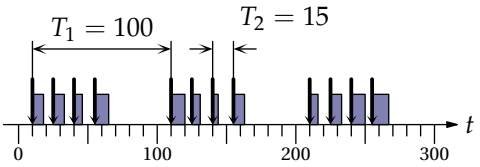
(a) Tarea periódica



(b) Tarea aperiódica



(c) Tarea esporádica



(d) Tarea esporádicamente periódica

Figura 1.2.: Ejemplos de diferentes tipos de tarea, atendiendo a la regularidad en sus instantes de llegada

consecutivas. Este tipo de tarea es típico de los sistemas multimedia o transaccionales [Tindell *et al.*, 1994]

- **Tareas con precedencia**. Este tipo de tareas suelen aparecer en sistemas distribuidos o multiprocesadores, en los que varias tareas cooperan para producir un resultado. En este caso, algunas de las tareas requieren como entrada los resultados producidos por otras tareas previas, por lo que no pueden comenzar su ejecución hasta que hayan finalizado las que las preceden. Los instantes de llegada de estas tareas, por tanto, dependen de los instantes de finalización de sus predecesoras. Esto hace que en general no sean periódicas (e incluso que su instante de llegada sea desconocido de antemano) lo que complica el análisis.

Además de la anterior clasificación de las tareas según sus instantes de activación, se pueden tener en cuenta otros aspectos que también influirán en el análisis de la planificabilidad, como son:

- Las tareas pueden acceder a **recursos compartidos**, lo que implica que deben sincronizarse de alguna forma para evitar acceder a la vez a estos recursos (esto es, para garantizar la exclusión mutua durante este acceso). Uno de los mecanismos más usados clásicamente en los sistemas operativos es

el del *semáforo*. No obstante, un bloqueo basado en semáforos "convencionales", plantea problemas en los sistemas de tiempo real. Básicamente se requiere que el tiempo que una tarea puede permanecer bloqueada a la espera del recurso ha de estar acotado, y esto no lo garantiza un semáforo "normal", por lo que es necesarios inventar otros mecanismos [Sha *et al.*, 1990].

- Los instantes de llegada de las tareas no son tan precisos como se supone en los modelos teóricos. En ocasiones, la activación de la tarea se retrasa con respecto al instante teóricamente esperado, debido a la granularidad del reloj que activa al planificador, o debido a la espera de mensajes de sincronización que no son instantáneos. La diferencia entre el instante en que teóricamente se debía activar la tarea y el instante en que realmente se activa se denomina ***jitter***. Este parámetro puede tenerse en cuenta en el análisis, para lo que se requiere que esté acotado por un valor máximo que usualmente se denota por J.

- El conjunto de tareas a planificar puede no ser fijo a lo largo del tiempo. Hay casos en los que las tareas se crean o destruyen en función de lo que ocurra en un entorno cambiante, de forma desconocida a priori. Este problema es estudiado en la planificación dinámica. En otros casos, si bien el entorno es cambiante, se conocen de antemano los posibles "estados" del sistema, y se conoce también con precisión las tareas que son necesarias en cada uno de esos estados. En este caso las tareas cambian "en bloque", cuando se pasa de un estado a otro. Se dice de este tipo de sistemas que presentan "**cambios de modo**" [Fohler, 1993].

Por otro lado, como ya se ha dicho, las tareas de tiempo real tienen restricciones temporales que cumplir. Éstas suelen expresarse como un plazo (o *deadline*), relativo al instante de activación de la tarea, que se denota por D y especifica el tiempo máximo que puede transcurrir desde que la tarea realiza la petición del procesador, hasta que se completa su trabajo (es decir, hasta que ha recibido todo el tiempo de computación solicitado). En las tareas periódicas, se suele comparar el plazo de la tarea con su periodo, lo que da lugar a tres posibilidades:

- **Plazos iguales a los periodos** ($D = T$). Este caso es típico de los sistemas de control, pues la señal de control que la tarea debe computar debe estar lista antes de adquirir la siguiente muestra. Este fue el primer caso en ser analizado y estudiado [Liu y Layland, 1973].

- **Plazos menores que los periodos** ($D < T$). Estudiado por primera vez por [Leung y Whitehead, 1982], es la primera extensión evidente al caso anterior.

- **Plazos arbitrarios** (pueden ser mayores que los periodos). Este caso es más complejo, puesto que una activación de la tarea puede ocurrir mientras la anterior aún está en ejecución [Tindell *et al.*, 1994].

Por otro lado, para las tareas no periódicas, cabe separar entre las aperiódicas, las cuales al no tener restricciones entre sus instantes de llegada, tampoco pueden obtener garantías de tiempo real, y las esporádicas. Para las aperiódicas, lo que se busca normalmente es conseguir un tiempo medio de respuesta lo más bajo posible (sin comprometer los plazos de las tareas periódicas o esporádicas), y no suelen llevar asociado un plazo. Las esporádicas en cambio sí pueden tener asociado un plazo, el cual puede compararse con el tiempo mínimo entre activaciones (suele ser menor). El análisis clásico trata a este tipo de tareas como periódicas, de cara a estudiar si cumplirán o no sus plazos [Tindell *et al.*, 1994].

Las tareas con relaciones de precedencia, como se ha dicho antes, suelen "colaborar" para producir un resultado. Por tanto no es habitual que cada una de ellas tenga un plazo individual, sino que más bien el plazo se establece desde el instante en que la primera tarea de la secuencia solicita el procesador hasta el instante en que la última tarea de la secuencia finaliza. Es lo que se denomina un plazo "extremo a extremo" o *end to end* [Sun, 1997].

1.2.3. Algoritmos de planificación

El algoritmo de planificación es el encargado de decidir qué tarea debe recibir cada recurso en un momento dado. Si no hay tareas que hayan solicitado el recurso (o que lo hayan solicitado pero estén pendientes de alguna sincronización), el recurso puede quedar ocioso. Si más de una tarea ha solicitado el recurso y cumple las condiciones de sincronización, el algoritmo debe decidir cuál de ellas lo recibirá. Incluso, si una tarea solicita el recurso, pero éste se halla ocupado por otra tarea, el algoritmo puede decidir expulsar a la que en ese momento tiene el recurso para asignárselo a la que acaba de pedirlo.

Un algoritmo de planificación "genérico" se diseñará de modo que maximice ciertos criterios. Así, por ejemplo, podría buscarse que todas las tareas reciban los recursos solicitados de forma equitativa, o que el tiempo promedio de finalización entre todas las tareas sea lo más bajo posible. En cambio, un algoritmo de planificación para sistemas de tiempo real tiene otro objetivo prioritario, y es el de ser analizable. Es decir, dado un conjunto de tareas y un algoritmo de planificación debe ser posible encontrar algún método analítico para responder a la pregunta "¿Cumplirán todas las tareas sus plazos?".

Existen muchos algoritmos de planificación, algunos son más sencillos de implementar en la práctica que otros, pero en general esta sencillez implica una infrautilización de los recursos (esto es, para garantizar que todos los plazos se cumplirán, el sistema debe estar poco utilizado por término medio). Esta variedad de algoritmos se pueden clasificar en grupos, atendiendo a diferentes criterios.

Atendiendo al conocimiento apriorístico de las tareas

Según se conozcan de antemano o no las características de las tareas a planificar, será posible realizar esta planificación *a priori*, o no quedará más remedio que planificar sobre la marcha, según llegan las tareas. Esto da lugar a dos familias de planificadores.

- **Estáticos**. Las características de todas las tareas que se ejecutarán en el sistema son conocidas de antemano. Esto permite al algoritmo garantizar (en caso que sea posible) que todas las tareas cumplirán sus plazos, o bien determinar si esto no es posible. Este análisis puede hacerse antes de construir el sistema, o bien mientras el sistema está funcionando. Esto da lugar a la siguiente sub-clasificación:

  - **Fuera de línea** (*offline*). El algoritmo de planificación se ejecuta antes de construir el sistema. Recibe información sobre las características de todas las tareas que componen el sistema, y genera como salida una tabla en la que indica en qué instante deberá concederse cada recurso a cada tarea, de forma que se garantice que todas cumplan sus plazos. Cuando el sistema se construya, habrá un proceso planificador que consultará esta tabla para ir concediendo a cada tarea los recursos.

    La principal ventaja de este enfoque es que, puesto que el análisis no se hace mientras el sistema funciona, hay tiempo suficiente para realizarlo. Esto permite utilizar técnicas computacionalmente costosas, encaminadas a la obtención de la planificación óptima, o casi óptima usando técnicas de *branch and bound* [Xu y Parnas, 1990; Peng y Shin, 1993]

  - **En línea** (*online*). El algoritmo de planificación se ejecuta como parte del propio sistema de tiempo real. Esto obliga a utilizar algoritmos mucho más simples, que por tanto pueden no estar usando la planificación óptima. A cambio, el sistema es más flexible, pues es muy simple realizar modificaciones de diseño en el mismo, como utilizar más tareas. Además, es posible utilizar los huecos en los que el planificador deja los recursos ociosos, para que estos recursos sean utilizados por otras tareas (por ejemplo aperiódicas).

- **Dinámicos**. Este tipo de algoritmos de planificación desconoce a priori las características de las tareas que llegarán. Cada vez que una nueva tarea se active, recibirá información en ese momento sobre sus características y plazos, y deberá rehacer una planificación que tenga en cuenta esta nueva tarea junto con las que ya tenía. Es evidente que este tipo de algoritmos incurrirán en una mayor sobrecarga en tiempo de ejecución que los estáticos, por lo que sólo se aplicarán a los casos en que efectivamente no se disponga de información a priori sobre las características de las tareas a planificar. Es

decir, para sistemas ubicados en entornos dinámicos o impredecibles. Naturalmente en sistemas así el análisis de planificabilidad también se verá mermado en su capacidad de garantizar el cumplimiento de los plazos. En este sentido se tienen dos clases de algoritmos:

- Algoritmos que "**hacen lo que pueden**" (*best effort*) por cumplir los plazos. Cada nueva tarea que llega, es aceptada. Posteriormente el algoritmo aplicará algún tipo de heurística sencilla para decidir qué tarea debe ejecutarse. Estas heurísticas suelen dar buenos resultados, pero no se ofrece garantía alguna de que las tareas vayan a cumplir sus plazos. Evidentemente este tipo de planificador no es adecuado para sistemas de tiempo real estrictos o "duros".
- Algoritmos con **garantías**. Cada vez que una nueva tarea llega al sistema, el planificador realiza un *test* de planificabilidad para decidir si aceptarla o no. Si el resultado del test le indica que, incluso aceptando la nueva tarea, hay suficientes recursos para que todas puedan cumplir sus plazos, la aceptará. En caso contrario la rechazará. De este modo el planificador garantiza que, al menos para las tareas que acepta, los plazos se cumplirán. Este tipo de planificador presenta una mayor sobrecarga en tiempo de ejecución que el planificador "que hace lo que puede".

Atendiendo a otros criterios

El término "desalojo" (o expulsión, en inglés *preemption*) alude al hecho de que una tarea que ya ha recibido el recurso y lo está utilizando, puede ser desalojada (expulsada) para darle el recurso que ocupaba a otra tarea. Esto puede ayudar a mejorar la planificabilidad. Según esta posibilidad esté permitida o no tendremos:

- Planificadores **sin desalojo** (*non-preemptive*). Una vez que una tarea ha conseguido un recurso, el planificador no podrá quitárselo. Dispondrá de él hasta que no lo necesite más. En el caso más común en que el recurso sea el procesador, esto significa que una vez que la tarea ha comenzado su ejecución, no podrá ser desalojada hasta que haya finalizado. Aunque este tipo de planificación hace más difícil que todas las tareas consigan sus plazos, evitan los problemas en los accesos a recursos compartidos.
- Planificadores **con desalojo** (*preemptive*). La tarea puede ser desalojada de un recurso, aún cuando no ha terminado de usarlo. A veces, incluso bajo este esquema de planificación, se permite a las tareas que tengan "regiones" de las que no pueden ser expulsadas. Esto complica el análisis de planificabilidad.

Otro concepto muy ampliamente utilizado es el de **prioridad**. La prioridad es un número que se asocia con cada instancia de cada tarea (es decir, se asocia con cada trabajo), y que indica de alguna forma su "urgencia" por terminar. El problema de asignar prioridades a las tareas puede ser resuelto en línea o fuera de línea. Un planificador basado en prioridades sería muy simple, puesto que cada vez que una tarea llega le basta con ponerla en una cola de tareas "listas", junto con el valor de su prioridad. Si es un planificador sin expulsión, esperará a que la tarea en ejecución finalice y después pasará a ejecutar la que tenga prioridad más alta en la cola de tareas listas. Si es un planificador con expulsión, en el mismo instante en que llega la nueva tarea la pondrá a ejecutar, si tenía mayor prioridad que la que estaba ejecutándose, la cual será momentáneamente expulsada (recuperará el procesador más tarde, cuando sea de nuevo la de mayor prioridad entre todas las tareas listas).

La prioridad puede asignarse a cada instancia de la tarea, en el momento en que esta se activa. Por ejemplo, la bien conocida política de asignación EDF (*Earliest Deadline First*, o "el plazo más temprano primero") asigna las prioridades de todas las tareas en ejecución cada vez que llega una nueva, de modo que la de mayor prioridad sea la que tiene el plazo absoluto más cercano [Liu y Layland, 1973]. En este caso mayor prioridad indica mayor urgencia.

La prioridad también puede ser asignada "fuera de línea", y en este caso lo habitual es asignar la misma prioridad a todas las instancias de una misma tarea. Es decir, se asignan las prioridades "por tareas" en lugar de "por trabajos". Ejemplos de este caso son las bien conocidas políticas RM (*Rate Monotonic*) y DM (*Deadline Monotonic*). En el primer caso [Liu y Layland, 1973], las tareas se ordenan en función de sus periodos de activación, y se asignan las prioridades en ese orden, de modo que la de menor periodo recibe la prioridad más alta. Las prioridades, por tanto, crecen monótonamente con la frecuencia de llegada, de ahí el nombre *Rate Monotonic*. El caso DM es análogo, pero las tareas se ordenan según sus plazos relativos en lugar de sus periodos. La tarea con plazo relativo más corto, recibe la mayor prioridad [Leung y Whitehead, 1982]. Observar la diferencia con EDF, que usa el plazo absoluto en lugar del relativo. Se ha demostrado que estos algoritmos son óptimos (RM para el caso en que los plazos sean iguales a los periodos, y DM para el caso general), en el sentido de que si estos algoritmos no pueden encontrar una asignación de prioridades que garantice que los plazos se cumplen, ningún otro algoritmo podrá encontrar esta asignación.

Finalmente, en caso de que sea necesario planificar varios recursos (varios procesadores), los algoritmos de planificación también se pueden clasificar en **locales** o **globales**. En la planificación local, una vez que una tarea ha sido asignada a un recurso (procesador) concreto, se ejecutará siempre en él. Por el contrario, en la planificación global cada instancia de la misma tarea podría ir a procesadores diferentes. Incluso una instancia en ejecución podría ser expulsada de un procesador para continuar ejecutándose en otro.

1.2.4. Análisis de planificabilidad

Una vez se han definido todas las tareas que componen el sistema, los recursos a administrar, y el algoritmo que se utilizará para planificarlos, se plantea la pregunta ¿se cumplirán todos los plazos de todas las tareas cuando sean planificadas con este algoritmo sobre estos recursos? Encontrar la respuesta a esta pregunta es uno de los objetivos básicos del análisis de planificabilidad. Pero otro objetivo también debería ser el encontrar fórmulas que relacionen los diferentes parámetros del sistema con la planificabilidad del mismo, de modo que la respuesta no sea un simple "si/no", sino que, en caso de que no sea planificable, se proporcione una intuición acerca de las causas, que permitan modificar el diseño para tomar las medidas correctoras adecuadas.

En la literatura se encuentran fundamentalmente las siguientes técnicas para el análisis de planificabilidad:

- Técnicas basadas en el **factor de utilización** del sistema Liu y Layland [1973]; Lehoczky [1990]. El factor de utilización de un recurso es la fracción de tiempo que permanece ocupado. En estas técnicas, en función del algoritmo de planificación que se esté usando, se tiene un "factor de utilización máximo" que puede alcanzarse sin comprometer el cumplimiento de los plazos. Por ejemplo, para el caso en que el algoritmo sea basado en prioridades y éstas se asignen con EDF, el factor de utilización máximo será 1 [Liu y Layland, 1973]. Es decir, el procesador puede estar ocupado el 100 % de su tiempo, y aún así los plazos se cumplirán. Si las prioridades se asignan mediante RM o DM, el factor de utilización máximo será inferior a 1, el valor exacto depende del número de tareas [Liu y Layland, 1973; Lehoczky *et al.*, 1989].

  El test de planificabilidad para estas técnicas es trivial. Basta calcular la utilización máxima que tendrá el sistema bajo análisis (el cual se calcula como la suma de todos los C_i/T_i, siendo C_i el peor tiempo de ejecución de cada tarea y T_i su periodo) y comparar ésta con la utilización máxima que permite el algoritmo de planificación. Si está por debajo, se garantizan todos los plazos. Si está por encima no se garantizan, lo cual no significa que vayan a perderse plazos, simplemente que no se puede saber si se perderán o no atendiendo solamente a la utilización [Lehoczky *et al.*, 1989]; sería necesario conocer más información sobre el sistema para decidirlo.

  La ventaja de estos métodos es la simplicidad de su test.

- Técnicas basadas en el **tiempo de respuesta** [Tindell *et al.*, 1994]. Estas técnicas se centran en encontrar el mayor tiempo de respuesta que puede tener cada tarea. Comparando este tiempo con el plazo relativo de cada tarea se puede comprobar si se cumplirán los plazos o no. Si esto se repite para cada tarea y en todas se cumple, el sistema se clasifica como "factible".

Para encontrar el mayor tiempo de respuesta posible se plantea siempre un escenario de "peor caso", en el cual los instantes de activación de las tareas coinciden de la peor forma posible, y los tiempos de procesador que solicitan las tareas son además los mayores entre todos los que podrían solicitar. Este escenario es muy pesimista, ya que hay muy pocas probabilidades de que aparezca en la práctica. No obstante, garantizando que incluso para ese caso se cumplen todos los plazos, se demuestra que el sistema cumplirá siempre sus plazos.

El coste computacional de esta técnica es muy bajo, ya que no necesita analizar todas las posibles combinaciones de activaciones de las tareas, sino tan solo el hipotético "peor caso" en que todas llegan en el mismo instante.

- Técnicas de **simulación**. La idea en este caso es construir un sistema sintético (o un modelo del mismo) que se comporte como el real, y "ejecutarlo", observando si se viola algún plazo de alguna de las tareas. La ventaja de este método es que es aplicable a cualquier tipo de tareas, sin restricciones. El problema es que no ofrece garantías 100 %. ya que es posible que durante el tiempo que duró la simulación ninguna tarea perdiera su plazo, simplemente porque la probabilidad de que tal cosa ocurra es muy baja (como suele serlo de hecho).

  Si bien la simulación no es un método fiable para garantizar si un sistema cumplirá o no sus restricciones de tiempo real, sí que ha sido utilizada en muchos trabajos para obtener datos estadísticos sobre los tiempos medios de respuesta, especialmente en el contexto de tareas aperiódicas con plazos blandos.

- Técnicas **estocásticas**. Estas técnicas parten de un modelo estadístico de cada tarea, en el cual no sólo se tiene en cuenta el peor caso posible, sino el "perfil de ejecución" de la tarea, es decir, todos los posibles tiempos de respuesta que puede presentar y la frecuencia (o probabilidad) con que aparece cada uno. Los instantes de llegada de las tareas también suelen ser aleatorios, y caracterizados mediante algún tipo de distribución estadística. En base a esta información estas técnicas intentan obtener la probabilidad de que la tarea cumpla o viole su plazo.

  Puesto que esta tesis se enmarca dentro de este último tipo de técnicas, se dedicará un capítulo completo ("Trabajo relacionado", página 21) a discutir con detalle las diferentes técnicas existentes en este campo y sus limitaciones.

1.3. El problema de la planificación en el ámbito de esta tesis

Una vez enunciado el problema del que se ocupa la disciplina del "tiempo real" en general, y enumerados todos los factores que influyen sobre este problema, centraremos la atención sobre un tipo de sistemas concreto y una forma particular de abordar su análisis, que serán los tratados en esta tesis.

De todas las posibilidades antes mostradas en la figura 1.1, eliminaremos gran parte de los factores para reducir el problema a tratar, y en particular nos quedamos con los aspectos siguientes (ver fig. 1.3, en la que se han subrayado los aspectos tratados):

- Arquitectura monoprocesador, en la cual el recurso a administrar es el tiempo de procesador.

- Tareas periódicas, o para ser más exactos, tareas con instantes de activación deterministas (el modelo que planteamos no se reduce a la periodicidad "clásica", sino que admite cualquier tipo de sistema en que los instantes de activación sean deterministas y sigan algún tipo de patrón repetitivo). No se consideran relaciones de precedencia entre las tareas. Inicialmente tampoco se considera el acceso a recursos compartidos ni la posible existencia de *jitter* en su activación. Posteriormente se ampliará el modelo para tomar en consideración estos dos aspectos.

  No se considera el caso de las tareas aperiódicas o esporádicas, en las cuales los instantes de llegada no son conocidos a priori, pero sí se ofrecen unas ideas acerca de cómo este tipo de tareas podría ser integrado en el análisis.

  Los plazos pueden tener cualquier valor, o incluso no estar especificados. En cualquier caso, el resultado del análisis será una función de probabilidad para cada tarea, en la que se encuentran todos los posibles valores que puede tomar el tiempo de respuesta y la probabilidad de cada uno de ellos. Por comparación de esta función con un plazo arbitrario se puede obtener la probabilidad de que se cumpla o pierda dicho plazo.

- Algoritmo de planificación basado en prioridades, según el cual el planificador garantiza en todo momento que la tarea en ejecución es la de mayor prioridad entre todas las tareas listas. No nos incumbe la política de asignación de prioridades, con tal de que sea determinista y siga un patrón repetitivo (al menos en las prioridades relativas de las tareas). Esto lo hace aplicable a las típicas políticas de asignación RM (o DM) y EDF. El planificador puede expulsar a una tarea cuando llega otra de mayor prioridad. También es posible (incluso más sencillo) analizar un planificador sin expulsión.

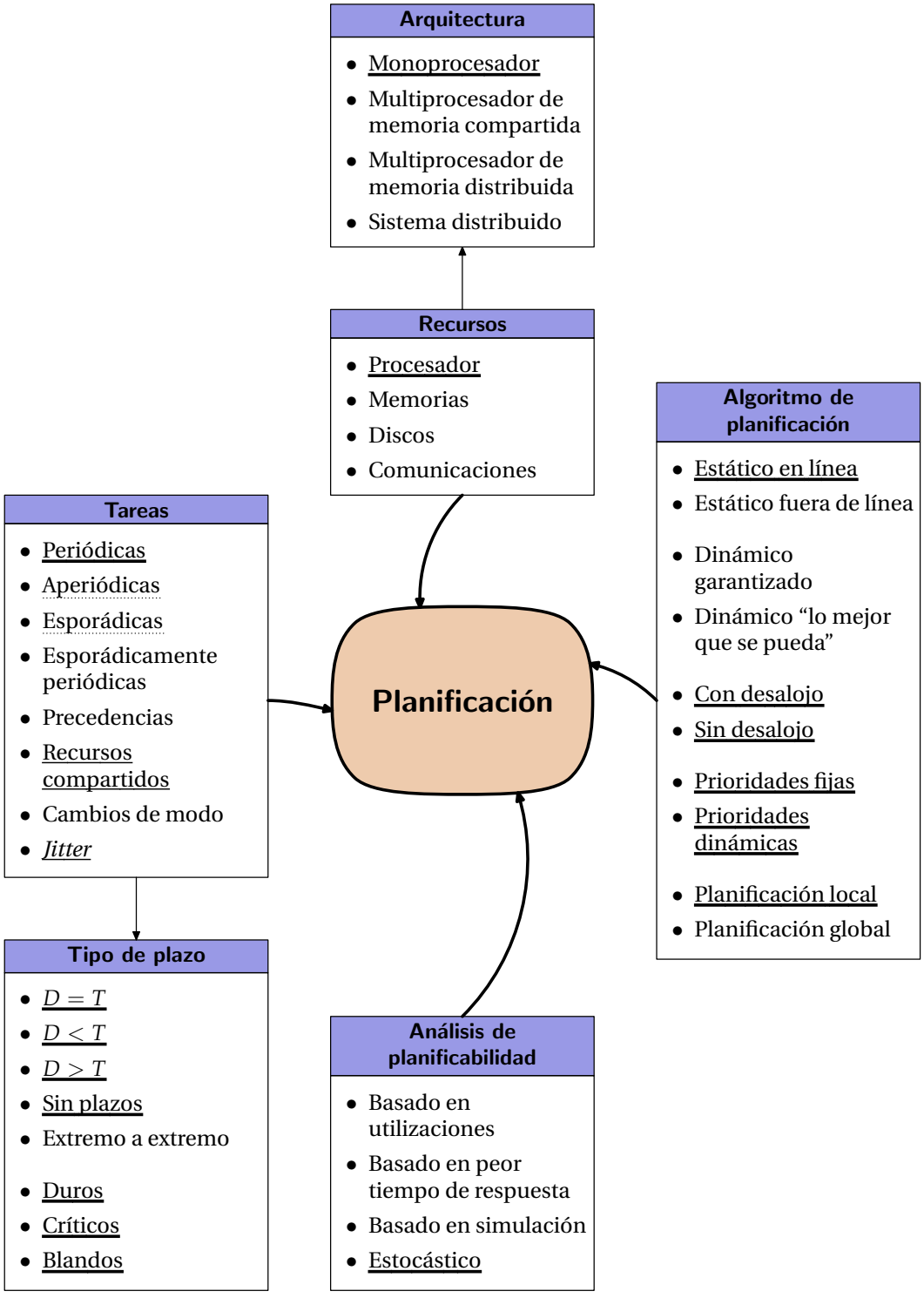


Figura 1.3.: Aspectos de la planificación en los sistemas de tiempo real tratados en el ámbito de esta tesis. Subrayados en línea gruesa los aspectos incluidos en el modelo básico. En línea fina los tratados en extensiones. En línea de puntos los tratados sólo a nivel descriptivo.

- El enfoque seguido para analizar la probabilidad será el estocástico. Asumiremos que los tiempos de ejecución de las tareas son variables aleatorias cuya distribución es conocida de antemano (pudo obtenerse por medida directa, por análisis estático del código, o por una mezcla de ambos, como por ejemplo en [Bernat *et al.*, 2002]). Nos planteamos encontrar la probabilidad *exacta* de que cada tarea cumpla o pierda su plazo.

1.4. Objetivos de esta tesis

Plantearemos un modelo de sistema en el que los instantes de activación de cada tarea se asumen conocidos de antemano de forma determinista, pero los tiempos de ejecución de cada tarea se modelarán como variables aleatorias independientes, caracterizadas por sus distribuciones de probabilidad. No se impondrá restricción alguna sobre la "forma" de dicha distribución, pero se asume conocida de antemano.

En el capítulo siguiente revisaremos las publicaciones que han atacado este mismo problema o similar. Veremos que cada uno de los trabajos existentes plantea algún tipo de restricción que simplifique el análisis. En esta tesis nos proponemos eliminar la mayor parte de estas restricciones, a la vez que plantear un modelo lo bastante genérico como para ser utilizable en diferentes situaciones.

En particular, los objetivos que nos planteamos son:

- Definir un modelo de sistema de tareas periódicas, en el que los tiempos de ejecución puedan ser aleatorios, que no imponga restricciones sobre los plazos de las tareas, que no requiera un planificador especial en su implementación, sino que use el clásico planificador basado en prioridades (RM, DM o EDF), que admita expulsión.

- Resolver de forma exacta el modelo anterior, esto es, encontrar las probabilidades de cada posible tiempo de respuesta de cada tarea. Esto requerirá:

  - Identificar bajo qué condiciones el modelo tiene solución, demostrando matemáticamente su existencia.
  - Diseñar operadores y algoritmos para trabajar con las funciones de probabilidad, que permitan obtener la respuesta buscada, demostrando matemáticamente la corrección de los mismos.
  - Idear formas aproximadas de encontrar la solución, para el caso en que la solución exacta sea demasiado costosa de obtener.

- Extender el modelo anterior para que pueda manejar también el acceso a recursos compartidos y el *jitter*. Esto requerirá probablemente renunciar al

análisis exacto, y tener que recurrir a aproximaciones. En este sentido será necesario

  - Idear un comparador de variables aleatorias que nos permita decir cuándo una variable es "peor" que otra en el sentido de que causaría mayores probabilidades de pérdidas de plazo.
  - Transformar el modelo extendido en un modelo más simple, mediante aproximaciones que garanticen que la solución obtenida por aproximación es más conservadora que la que se obtendría resolviendo de forma exacta el modelo extendido, es decir, que produce tiempos de respuesta "peores" en sentido estocástico.

- Estudiar teóricamente la complejidad del método propuesto, identificando qué parámetros del sistema influyen en el coste de resolverlo. Realizar experimentos para medir el coste real de analizar diferentes sistemas, en función de los parámetros identificados.
- Comparar los resultados obtenidos mediante nuestro análisis frente a otros métodos preexistentes que persigan el mismo propósito.

2. Trabajo relacionado

> If you steal from one author, it's plagiarism; if you steal from many, it's research.
>
> *(Wilson Mizner)*

En este capítulo detallaremos los trabajos que abordan problemas similares al que se trata en esta tesis. Veremos que cada enfoque tiene sus limitaciones y su campo de aplicación, y llegaremos a la conclusión de que ninguno de ellos resuelve exactamente el mismo problema que el resuelto en esta tesis.

En las explicaciones de los modelos utilizados y de los métodos seguidos para analizarlos, inevitablemente se han de utilizar términos y conceptos que aún no han sido definidos en esta tesis, (como el *hiperperiodo*, la *función de probabilidad*, la *cadena de Markov*, etc.) Si el lector no está familiarizado con estos conceptos, puede preferir posponer la lectura de este capítulo hasta después del capítulo 5.

2.1. Análisis clásico del tiempo de respuesta

Como ya se explicó en la introducción, uno de los enfoques clásicos para determinar si una tarea cumplirá o no su plazo consiste en calcular cuál sería su tiempo de respuesta en el peor caso posible (esto es, el tiempo transcurrido desde que la tarea se "activa" o está lista para ejecutarse hasta el instante en que finaliza su ejecución), y comparar este tiempo con su plazo [Tindell *et al.*, 1994]. Evidentemente, si es menor, se tendrá la garantía de que siempre se cumplirá el plazo. En cambio si es mayor, tan sólo se sabrá que hay al menos un caso en que no se cumple, pero no se sabrá si ese caso es común o por el contrario es rarísimo. En otras palabras, no se tiene información sobre la *probabilidad* de que este peor caso aparezca.

Si una tarea se ejecutara sola en el sistema, sin interferencias de otras tareas, aún así su tiempo de ejecución sería variable, pues dependiendo de los datos de entrada con los que tenga que trabajar, requerirá más o menos tiempo de procesador. Si además se tienen en cuenta otros efectos de la arquitectura (como el efecto de la memoria *cache*, o de la segmentación del cauce de ejecución) se tendrán más factores que harán variar el tiempo de ejecución de la tarea. Este tiempo que la tarea requeriría estando sola en el sistema, es lo que se denomina *tiempo*

de ejecución[1] (no confundir con el tiempo de respuesta que es el que presentará realmente una vez que tenga que competir con las restantes tareas por el uso de los recursos).

Para que la tarea sea analizable desde el punto de vista del tiempo real, su tiempo de ejecución máximo debe estar acotado y debe ser conocido. En general no es sencillo determinar cuál será el peor tiempo de ejecución de una tarea. Puede obtenerse por medida directa, o mediante análisis estático del código, o mediante técnicas mixtas. Hay toda una línea dentro del campo de tiempo real dedicada a este problema, denominado *Worst-Case Execution Time Analysis* (o análisis del *WCET*).

Asumiendo que los peores tiempos de ejecución son conocidos para todas las tareas del sistema, el análisis de tiempo de respuesta clásico planteará un escenario de peor caso en el que:

1. Todas las instancias de todas las tareas van a requerir su tiempo de ejecución máximo.

2. Las activaciones de las tareas se presentan de tal forma que causan la máxima interferencia a la tarea que se está analizando. Se dice que esta tarea se activó en su *instante crítico*.

Encontrar el instante crítico no es evidente. Si los plazos de las tareas son iguales o inferiores a sus periodos, se ha demostrado [Liu y Layland, 1973] que el instante crítico ocurre cuando la tarea bajo consideración se activa a la vez que todas las demás tareas de mayor prioridad. En cambio, si los plazos de las tareas pueden ser superiores a sus periodos el escenario cambia, ya que cabe la posibilidad de que cuando una tarea se active, aún no haya terminado de ejecutarse la anterior instancia de esa misma tarea, y por tanto experimente un retraso mayor del que sufriría cuando todas las tareas se activan a la vez. Para este segundo caso se ha demostrado [Lehoczky, 1990] que el instante crítico ocurre en alguna de las activaciones que tienen lugar dentro de un "periodo ocupado" (*busy period*), sin que sea posible determinar en cuál de ellas. Un "periodo ocupado" comienza en el instante en que todas las tareas del sistema llegan a la vez, y termina cuando todo el trabajo solicitado a lo largo del periodo ocupado ha sido servido.

Así pues, para el caso de plazos iguales a periodos, basta con estudiar cuál sería el tiempo de respuesta de la primera activación de una tarea, suponiendo que todas las demás llegan a la vez y que todas requieren su máximo tiempo de ejecución. En cambio, para el caso de plazos mayores que periodos es necesario estudiar el tiempo de respuesta de *cada una* de las activaciones de la tarea que tienen lugar dentro de un "periodo ocupado", y quedarse finalmente con la mayor de todas ellas. Los periodos ocupados tienen longitud finita siempre que la utilización

[1] Se trata del parámetro que en teoría de colas se denomina usualmente "tiempo de servicio".

total del sistema sea menor que 1, por lo que incluso en este segundo caso el número de instancias a analizar está acotado y el método finalizará en tiempo finito. Si en cambio la utilización es mayor que 1, el periodo ocupado tendría longitud infinita, por lo que el sistema no sería analizable. De todas formas, un sistema con utilización mayor que 1 no sería planificable bajo este enfoque, puesto que estaría exigiendo al procesador más tiempo del disponible.

El problema del planteamiento anterior es que las tareas no requieren siempre su tiempo de ejecución máximo. De hecho es muy común que el tiempo de ejecución máximo tenga una probabilidad muy baja, y además esté muy alejado del tiempo de ejecución promedio. En este sentido, el análisis clásico es muy pesimista, puesto que está suponiendo que todas las tareas van a requerir su peor tiempo. La probabilidad de este evento sería el producto de las probabilidades del peor tiempo para cada una de ellas, y por tanto mucho más baja aún. Además, asume un "instante crítico" para cada una de las tareas, y es posible que tal instante crítico nunca aparezca en la realidad (por ejemplo, si las tareas tienen desplazamientos u *offsets*), o que aparezca sólo ocasionalmente (una vez por hiperperiodo).

Debido a este pesimismo, se han investigado nuevos modelos y métodos de análisis que permitan incorporar información no sólo sobre los tiempos de ejecución máximos de las tareas, sino también de otros tiempos de ejecución que las tareas puedan presentar.

2.2. Una taxonomía de los trabajos que asumen tiempos de ejecución variables

Hay una serie de trabajos que asume que los tiempos de ejecución de las tareas no quedan bien representados por un único valor (peor tiempo), y que defienden la necesidad de modelar el tiempo de ejecución de cada tarea mediante un conjunto de valores. No obstante, partiendo de este supuesto inicial, la forma de modelar y resolver el problema difiere de unos autores a otros.

En el cuadro 2.1 se muestran los trabajos más relevantes que han abordado el problema de los tiempos de ejecución variables. Las referencias están por orden cronológico, la última de las cuales es esta misma tesis.

Los diferentes factores que se muestran en el cuadro2.1 significan lo siguiente:

- **Tareas**. El instante en que cada tarea se activa puede ser conocido de forma determinista (típicamente en caso de tareas periódicas), o puede ser aleatorio (tareas aperiódicas)[2]

[2] También se podría haber incluido en la taxonomía si, en el caso de tareas periódicas, el modelo tiene en cuenta la existencia de un *offset* en la primera activación de cada tarea, o si permite la

	Tareas					Plazos				Planificador					Solución		
	Periódicas	Aperiódicas	Rel. precedencia	Interbloqueo	$U^{\max} > 1$	$D < T$	$D = T$	$D > T$	*End-to-end*	RM/DM	EDF	Específico	Sin expulsión	Con expulsión	Exacta	Conservadora	Aproximada
Tia *et al.* [1995]	■	□	□	□	■	■	■	□	□	■	□	□	□	■	□	□	■
Lehoczky [1996, 1997]	□	■	□	□	■	■	■	■	□	□	■	□	□	■	□	□	■
Mok y Chen [1997]	■	□	□	□	□	■	■	□	□	■	□	□	□	■	□	■	□
Atlas y Bestavros [1998]	■	□	□	□	□	□	■	□	□	■	□	■	□	■	□	■	□
Kalavade y Moghê [1998]	■	□	■	□	□	□	■	□	■	■	■	□	■	□	■	□	□
Abeni y Buttazzo [1998, 1999, 2001]	■	■	□	□	■	■	■	■	□	□	□	■	□	■	□	□	■
Gardner y Liu [1999]; Gardner [1999]	■	□	□	■	■	■	■	■	□	■	□	□	□	■	□	■	■
Zhou *et al.* [1999]; Hu y Zhou [2001]	■	□	■	□	□	□	■	□	□	■	□	□	□	■	□	□	■
Manolache *et al.* [2001]; Manolache [2002]	■	□	■	□	□	■	■	□	■	■	■	□	■	□	■	□	□
Kim *et al.* [2002]	■	□	□	□	■	■	■	■	□	□	■	□	□	■	□	□	■
Esta tesis [Díaz, 2003]	■	□	□	■	■	■	■	■	□	■	■	□	■	■	■	■	■

Cuadro 2.1.: Algunos trabajos que tratan los tiempos de ejecución como variables. Enfoque y limitaciones de cada uno

Algunos modelos permiten expresar *relaciones de precedencia* entre las tareas que componen el sistema. En este tipo de modelos normalmente la "actividad" a desarrollar se compone de una secuencia de tareas, de modo que cada una debe esperar a que finalice la anterior para poder comenzar su ejecución. Normalmente sólo la primera tarea de esta secuencia tiene una activación periódica, y el plazo suele estar asociado a la secuencia completa, en lugar de a cada tarea particular.

El *interbloqueo* hace referencia a la posibilidad de que las tareas accedan a recursos compartidos que deban ser protegidos con semáforos u otros mecanismos que garanticen la exclusión mutua. La existencia de estos mecanismos puede causar que una tarea de alta prioridad no pueda ejecutarse porque necesita un recurso que está siendo retenido por una tarea de prioridad más baja. Esta situación complica el análisis.

Finalmente, la columna $U^{\max} > 1$ indica si el modelo permite analizar sistemas en los que la suma de utilizaciones máximas de todas las tareas exceda la unidad. Un sistema así no sería factible bajo un análisis clásico, sin

aparición de *jitter* (un ligero retardo entre el instante teórico de activación y el instante real). Por no extender más la tabla se han omitido estos detalles, ya que además prácticamente ninguno de los trabajos los tiene en cuenta, salvo la presente tesis.

embargo bajo un análisis estocástico tiene mucho sentido analizar este tipo de situaciones, ya que la probabilidad de que todas las tareas requieran a la vez su tiempo máximo es muy baja. El análisis se complica extraordinariamente para este caso, debido a que existe una probabilidad no nula de que al final de un hiperperiodo quede trabajo sin servir. Esto invalida la hipótesis de que los hiperperiodos son independientes entre sí, hipótesis admitida implícitamente por unos cuantos de los trabajos examinados.

- **Plazos**. Algunos de los trabajos estudiados imponen la restricción de que los plazos deban ser inferiores a los periodos, junto con una política de eliminación de las tareas que no cumplen sus plazos. Esto simplifica el análisis porque garantiza que cuando una tarea nueva es activada, no hay ninguna otra instancia previa de la misma tarea ejecutándose en el sistema. Bajo esta hipótesis, la peor activación posible sería la que ocurre en un instante crítico (instante en que todas las tareas del sistema se activan a la vez). Algunos de los trabajos utilizan esta hipótesis para centrarse en el estudio del caso crítico únicamente. Los enfoques más generales no imponen restricciones a los plazos, que por tanto pueden ser menores, iguales o mayores que los periodos.

  En los casos en que el modelo permite expresar dependencias entre tareas, los plazos suelen ser del tipo "extremo-a-extremo" (*end-to-end*).

- **Planificador**. Todos los trabajos examinados utilizan un planificador basado en prioridades, siendo la política de asignación de prioridades bien RM o bien EDF. Algunos modelos no permiten la "expulsión", es decir, una vez que una tarea entra en ejecución, ya no será expulsada hasta que finalice, aunque puedan llegar otras de mayor prioridad. Esta limitación simplifica en gran medida el análisis, ya que el tiempo de respuesta de la tarea no se ve influido por las tareas que lleguen después.

  Finalmente algunos de los trabajos requieren un planificador específico, normalmente porque implementan algún tipo de "servidor", es decir, un proceso que se ocupa de asignar tiempo de procesador a las tareas que llegan en instantes arbitrarios, a la vez que controlan que no excedan una cierta cantidad de tiempo prefijado. De este modo se limita la influencia de unas tareas sobre otras.

- **Solución**. Una vez establecido el modelo y las hipótesis simplificadoras oportunas, los autores proponen diferentes métodos de análisis y buscan una solución que podríamos clasificar en tres categorías:

  - *Exacta*: Obtienen la probabilidad exacta de que cada tarea cumpla o pierda su plazo.

- *Conservadora:* (o pesimista). Obtienen una probabilidad de cumplir el plazo que, si bien no es exacta, se garantiza que siempre está por debajo de la probabilidad real. Este tipo de soluciones podría ser aplicable a sistemas de tiempo real firmes, en los que se esté dispuesto a admitir un porcentaje *máximo* de violaciones de plazo.
- *Aproximada:* Obtienen una probabilidad de cumplir los plazos que intenta ser lo más próxima posible a la real, pero no se garantiza que esté siempre por debajo. Este tipo de soluciones sólo sería útil para sistemas de tiempo real blando.

En cuanto a la forma de abordar el problema, cada autor plantea algún tipo de hipótesis o restricción que le permita simplificar el análisis. Podríamos catalogar estas simplificaciones en los tipos siguientes:

- Restricción del estudio a un "**instante crítico**". En lugar de analizar cada activación de cada tarea, se estudia únicamente una activación que tiene lugar en un instante crítico. Este es el enfoque seguido por Tia *et al.* [1995]; Gardner [1999]. La existencia de un instante crítico implica restringir el modelo a casos en que el plazo sea inferior al periodo [Tia *et al.*, 1995], o bien analizar todas las activaciones que tienen lugar dentro de un "periodo ocupado" [Gardner, 1999], si bien la validez de cualquiera de las hipótesis es cuestionable cuando $U^{\max} > 1$.
- Utilización de un **planificador especial** que aisle el efecto de unas tareas sobre otras. Este es el enfoque seguido por Abeni y Buttazzo [1998, 1999, 2001]; Atlas y Bestavros [1998]. Cada tarea puede analizarse independientemente de las demás mediante teoría de colas, modelando cada tarea como un cliente y el procesador como un servidor. La principal desventaja de este método es que no es válido para analizar sistemas planificados con algoritmos convencionales.
- Independencia entre los hiperperiodos. Si la **utilización máxima es inferior a 1**, cada hiperperiodo no puede afectar a los siguientes. Esto permite enumerar todos los posibles estados del sistema, entendiendo por estado el conjunto de tareas que están activas en un instante dado. Se pueden calcular las probabilidades de transición de un estado a otro y, aplicando teoría de procesos estocásticos y cadenas de Markov, esto permite hallar la probabilidad de que el sistema se halle en un estado cualquiera, y de ahí deducir las probabilidades de perder plazos. Este es el enfoque seguido por Kalavade y Moghê [1998]; Manolache [2002]
- **Tráfico intenso**. Bajo la hipótesis de que todas las tareas tienen aproximadamente el mismo comportamiento estocástico y que en promedio el sis-

tema está completamente utilizado (no tiene apenas tiempo libre), el comportamiento del sistema se puede aproximar mediante un proceso estocástico denominado "movimiento Browniano". Este es el enfoque seguido por Lehoczky [1996, 1997], que desarrolla una nueva teoría de colas con "ingredientes" de tiempo real.

Otros trabajos siguen un enfoque ligeramente diferente, que hace difícil clasificarlos según la taxonomía anterior, así por ejemplo, en [Mok y Chen, 1997] los tiempos de ejecución de las tareas no son aleatorios, pero tampoco son fijos, sino que siguen un **patrón repetitivo**. No se ofrecen probabilidades de cumplir plazos, sino un nuevo test de planificabilidad que indica si todas las tareas cumplirán sus plazos o no. En [Zhou *et al.*, 1999; Hu y Zhou, 2001] no se busca la probabilidad de que cada tarea individual cumpla un plazo, sino una **métrica global** que los autores denominan "probabilidad de que el sistema sea factible". En [Bernat *et al.*, 2002] se plantea un **análisis estocástico del *WCET***, es decir , no se analiza el tiempo de respuesta de las tareas, sino su tiempo de ejecución (el tiempo que tardarían asumiendo que están solas en el sistema), pero la técnica utilizada es la composición de pequeños trozos de código cuyo tiempo de ejecución es una variable aleatoria de distribución conocida, por lo que las técnicas estadísticas utilizadas son muy similares a las de esta tesis.

En el siguiente apartado se comentan con más detalle cada uno de estos trabajos y sus limitaciones.

2.3. Análisis estadístico del tiempo de respuesta

2.3.1. Asumiendo un instante crítico

En el trabajo de Tia *et al.* [1995] se plantea por primera vez la técnica de encontrar las probabilidades de perder plazos mediante una "traducción" directa del método clásico de cálculo del tiempo de respuesta. En el método clásico, el tiempo de respuesta se calcula como la suma del tiempo de ejecución de una tarea más la interferencia que sufre (que son los tiempos de ejecución de las tareas de mayor prioridad que pueden causarle expulsión). En el trabajo de Tia *et al.* los tiempos de ejecución son variables aleatorias de distribución conocida, y se asume independencia entre las distribuciones, de modo que lo que en el análisis clásico era una suma, se convierte aquí en una convolución de funciones de probabilidad.

De todas formas, este análisis no busca la distribución de probabilidad del tiempo de respuesta, sino la de la carga que debe ser servida para que la tarea bajo análisis pueda terminar. Esta carga se compone del tiempo de ejecución de la propia tarea, más los de las tareas que la vayan expulsando. En general, ya que la carga del sistema va variando con el tiempo (por llegar trabajos nuevos), su distribución estadística también irá variando. El análisis busca encontrar $w_i(t)$ que es la

distribución estadística de la carga en el instante t que debe ser servida para que finalice la tarea i. Una vez conocida esta distribución, se puede hallar la probabilidad de que la carga en el instante t sea menor que t. Esta será la probabilidad de que la tarea finalice en t. En particular, se puede buscar la probabilidad de que la carga sea menor que t para el instante que representa el plazo absoluto de la tarea. Esto dará una aproximación de la probabilidad de cumplir el plazo. La aproximación es pesimista, y puede refinarse si se encuentran probabilidades más altas de que $w(t) < t$ antes del instante que representa el plazo. El algoritmo propuesto es buscar estas probabilidades en los instantes de llegada de cada una de las tareas que podría expulsar a la que está siendo analizada.

Encontrar la distribución $w_i(t)$ no es evidente, y para poder aproximar el cálculo como una mera convolución de variables aleatorias, es necesario realizar una serie de simplificaciones y aproximaciones:

- Se asume que el peor tiempo de respuesta de la tarea bajo análisis ocurrirá en el instante crítico (t_0) en que todas las tareas llegan a la vez, y que en este instante crítico no hay carga pendiente en el sistema, es decir $w_i(t_0) = 0$. Para que esto sea así, el sistema debe tener plazos iguales o inferiores a los periodos, y utilización máxima menor que 1.

- La distribución de la carga $w(t)$ en el instante t no se encuentra de forma exacta, sino que se aproxima como la suma (convolución) de los tiempos de computación de todos los trabajos que llegan entre el instante crítico y t. Esto quiere decir que no se tiene en cuenta que la carga va siendo servida a medida que avanza el tiempo, y que eventualmente podría llegar a ser cero. En todo caso, la suma que se obtiene será siempre superior a la carga real, por lo que se está en un caso pesimista que es válido para extraer conclusiones de planificabilidad.

La primera simplificación restringe el campo de aplicabilidad del método, mientras que la segunda hace que las probabilidades halladas no sean exactas, sino simplemente límites pesimistas.

En su tesis doctoral, Gardner [1999] parte de las ideas de Tia *et al.* [1995] tratando de eliminar algunas de sus restricciones. El modelo del sistema es el mismo que en [Tia *et al.*, 1995], y lo que cambia es el método para obtener las probabilidades de pérdida del plazo.

Estas probabilidades no se encuentran de forma exacta, sino que se busca un valor que se garantice ser más pesimista que el real. Para ello Gardner también presupone un "instante crítico" en el que todas las tareas llegan a la vez, pero en lugar de analizar únicamente la activación que tiene lugar en ese instante, analiza todas las activaciones de la tarea que tienen lugar dentro del periodo ocupado

(o *busy-period*) que comienza en dicho instante crítico. Para cada activación, encuentra una probabilidad de pérdida de plazo diferente, y se queda con la peor de todas ellas.

El problema es que la longitud del *busy-period* es desconocida de antemano, pues es una variable aleatoria que depende de los tiempos de ejecución de las tareas. El método de Gardner, a la vez que va computando las probabilidades de perder el plazo, también va calculando la probabilidad de que el *busy-period* haya finalizado en el instante de llegada de cada tarea. Si esta probabilidad es 1, el *busy-period* habrá finalizado con certeza y ya no será necesario examinar más tareas.

Este enfoque tiene dos problemas importantes cuando la utilización máxima es mayor de 1 (es decir, cuando los peores tiempos de ejecución de las tareas exigen más tiempo del físicamente disponible). Cuando se da este caso (que por otra parte es el caso más interesante y el que Gardner se propone resolver), ocurre que:

- La longitud del *busy-period* es infinita. Esto se debe a que, para cualquier instante t, existe una probabilidad no nula de que todas las tareas hayan requerido su peor tiempo hasta ese instante, y por tanto una probabilidad no nula de que haya trabajo pendiente.

  Esto implica que el número de activaciones a analizar sería infinito. No obstante, Gardner ha detectado que, a medida que se van obteniendo las probabilidades de perder plazo para cada una de las activaciones de las tareas, y se va quedando con la más pesimista de ellas, el valor de esta probabilidad peor va "convergiendo" a un límite, que en teoría se alcanzaría en la última tarea del *busy-period* (que no se alcanza nunca en la práctica por ser infinito).

  Gardner en cambio no demuestra que esta convergencia siempre tenga lugar, o las condiciones bajo las cuales se dará, ni proporciona ninguna herramienta analítica para encontrar el valor límite de esta probabilidad.

- El *busy-period* que comienza cuando todas las tareas llegan a la vez, no contiene necesariamente la "peor activación" de la tarea. La demostración de que un instante así contiene la peor activación, requiere como una de las hipótesis que la utilización máxima sea menor que 1. En el momento que esta hipótesis no se cumple, la peor activación de la tarea puede ocurrir para otra combinación de fases.

  Gardner detecta este problema, pero aún así mantiene el análisis descrito bajo la suposición de que la probabilidad suministrada por su método es razonablemente buena. Hace la hipótesis de que el caso que produciría el peor tiempo de respuesta aparece muy infrecuentemente, y por tanto apenas afecta a la probabilidad total de perder el plazo.

> Si bien puede ser cierto que la probabilidad suministrada por su análisis sea una aproximación razonable a la verdadera probabilidad, lo cierto es que ya no se trata de un límite pesimista, y por tanto deja de ser válido como garantía para sistemas más estrictos.

2.3.2. Con planificadores específicos

En el trabajo de Abeni y Buttazzo [1999] se plantea un modelo en el que coexisten tareas de tiempo real duro, blando y tareas no de tiempo real. Las de tiempo real duro son caracterizadas como en el análisis clásico por su periodo (o mínimo intervalo entre llegadas), su peor tiempo de ejecución (*WCET*) y un plazo estricto que debe ser garantizado. Para las tareas de tiempo real blando, los tiempos de ejecución son variables aleatorias con distribuciones arbitrarias (pero conocidas), y los instantes de llegada también son aleatorios, siendo conocida la distribución de los intervalos entre llegadas. Para este tipo de tareas los plazos no son firmes, sino probabilísticos. Es decir, la "calidad de servicio" exigida a cada tarea viene dada en forma de un par (δ, p), y la tarea se considerará planificable si la probabilidad de que pierda su plazo δ es inferior a p. El sistema se considerará "factible" si todas las tareas son planificables con este criterio, y además se garantiza el cumplimiento de los plazos para las de tiempo real duro.

La estrategia utilizada para poder analizar un sistema así consiste en idear una planificación con *reserva de ancho de banda*, de modo que se asigna a cada tarea de tiempo real blando una fracción del tiempo de procesador y el planificador asegura que la tarea nunca excederá el tiempo que le ha sido asignado. Si la tarea agota el tiempo asignado sin haber finalizado, se la "suspende" temporalmente para dar paso a otras tareas. De este modo no se comprometen las garantías de calidad que se habían asegurado para otras tareas. El planificador utilizado es de tipo EDF, con un servidor para tareas de tiempo real blando del tipo "Servidor de ancho de banda constante", ideado por los mismos autores en un trabajo previo [Abeni y Buttazzo, 1998].

Asumiendo este tipo de planificador, se aisla el efecto de las tareas de tiempo real blando sobre las de tiempo real duro. Desde el punto de vista de éstas últimas, el servidor de tareas blandas puede analizarse como una tarea más, caracterizada por su "tiempo de ejecución máximo" (que es el tiempo "presupuestado" o reservado para las tareas blandas) y su periodo (que es el intervalo entre dos activaciones del servidor). Desde el punto de vista de las tareas de tiempo real blando, el efecto de unas sobre otras al competir por el procesador durante el tiempo que les asigna el servidor, puede ser modelado mediante teoría de colas (en la que los clientes serían las tareas de tiempo real blando y el servidor sería determinista). El análisis considera dos posibles casos: tiempos de ejecución aleatorios con periodos entre llegadas fijos, y tiempos de ejecución fijos con periodos entre llegadas

aleatorios. Para cada uno se desarrolla un análisis que permite encontrar las probabilidades de pérdida de plazos para las tareas blandas.

Posteriormente, los mismos autores extienden el modelo en otro trabajo [Abeni y Buttazzo, 2001],para cubrir el caso en que tanto los tiempos de computación como los intervalos entre llegadas son aleatorios, y además el planificador no es necesariamente el "servidor de ancho de banda constante", sino cualquier otro planificador con reserva que garantice el "aislamiento" entre tareas, de modo que cada tarea pueda ser analizada independientemente dentro de su "pedazo reservado de procesador".

La principal limitación de este análisis es que exige un tipo particular de planificador, y por tanto no es aplicable a los planificadores más comúnmente implementados, como los basados en prioridades RM, DM o EDF. Por otro lado, el análisis requerido para resolver el modelo de colas subyacente implica el buscar un vector propio de una matriz infinita, y el método utilizado para salvar este problema consiste simplemente en la truncación de esta matriz para trabajar con una matriz finita, sin especificar claramente el criterio para elegir el punto de truncación. Como veremos, en esta tesis encontraremos el mismo problema (hallar un vector propio de una matriz infinita), pero daremos una forma exacta de resolverlo.

Atlas y Bestavros [1998] idean un mecanismo de planificación nuevo, que denominan "Planificación RM estadística" (*SRMS*[3]) cuyo objetivo es implementar un control de admisión de las tareas que tienen alta variabilidad en sus tiempos de ejecución. Si el planificador acepta la tarea, el cumplimiento de su plazo estará garantizado. Si no puede garantizarlo, el planificador la rechaza. El análisis tiene dos partes, un test de admisión (que es muy simple, y se basa en el límite del factor de utilización y no tiene en cuenta los parámetros estocásticos de las tareas) y un análisis estadístico de la calidad de servicio. Este último se centra en calcular la probabilidad de que cada tarea sea aceptada o rechazada, y esta probabilidad se utiliza como una medida de la calidad de servicio (QoS) del sistema. Se dice que el sistema es planificable si proporciona la calidad de servicio que el diseñador desea.

El modelo de tareas es periódico, y se suponen las tareas ordenadas por orden de frecuencia, es decir, la tarea 1 tendrá el menor periodo de todas, y a medida que se avanza de una tarea a la siguiente los periodos van creciendo. Para cada tarea se define su "superperiodo" como un intervalo de tiempo que comienza cuando una instancia de la tarea llega, y tiene una longitud igual al periodo de la siguiente tarea. El diseñador/analista asignará a cada tarea τ_i un tiempo a_i (que los autores denominan *permiso*[4]). El planificador garantiza que en cada superperiodo la tarea τ_i tendrá disponible un tiempo igual a a_i para ejecutar sus diferentes

[3] *Statistical Rate Monotonic Scheduling*
[4] *allowance*

instancias. Los detalles de cómo funciona el planificador para proporcionar esta garantía pueden consultarse en [Atlas y Bestavros, 1998]. En el superperiodo llegan varias instancias de la tarea, y la probabilidad de que cada una de ellas resulte aceptada o rechazada depende de si se han aceptado o no las previas, ya que si se han aceptado el tiempo disponible para las siguientes disminuye. Así, la primera instancia será admitida si el tiempo que solicita es inferior a a_i. La segunda instancia en ese superperiodo será admitida si la primera fue rechazada y el tiempo solicitado por la segunda es inferior a a_i, o bien si la primera fue aceptada y el tiempo solicitado por ambas es inferior a a_i, etc.

Los autores encuentran una fórmula muy simple para determinar la probabilidad de que cada una de las tareas que pueden llegar en el superperiodo sea aceptada o rechazada. A partir de ellas la probabilidad total de que una tarea sea aceptada o rechazada será el promedio de éstas. Esta probabilidad depende de los valores que el analista haya dado a los "permisos" a_i de las tareas. Para encontrar esta sencilla fórmula deben realizar la hipótesis de que los periodos de las tareas son armónicos (es decir, cada periodo es múltiplo del periodo de la tarea anterior). En caso de que no sea así, el tratamiento se complica, pero por razones de espacio los autores no muestran el desarrollo ni las conclusiones para este caso.

Vemos pues que aunque se da un tratamiento probabilístico al problema, éste es bastante diferente de los otros trabajos estudiados y del que se plantea en esta tesis, puesto que los autores no buscan la distribución del tiempo de respuesta, ni siquiera la probabilidad de cumplir los plazos, sino la probabilidad de que las tareas sean aceptadas. Una vez que lo han sido, el planificador garantiza que cumplirán su plazo.

Las principales limitaciones de este método son:

- El planificador no es estándar, sino uno especialmente diseñado para este problema.
- Los plazos se consideran en todo momento iguales a los periodos.
- No se permiten utilizaciones mayores de 1. Si llega una tarea que va a solicitar más tiempo que su periodo, directamente es rechazada por el planificador.
- Se asume que el tiempo que va a necesitar la tarea es conocido por el planificador en el instante en que ésta llega, aunque se trata de una variable aleatoria. Este requisito parece poco verosímil.

2.3.3. Asumiendo independencia entre hiperperiodos

El trabajo de Kalavade y Moghê [1998] se centra en el análisis de servidores de contenidos multimedia, que se adapten automáticamente a las condiciones de su

entorno. Por ejemplo, si se detecta que el ancho de banda de la red es escaso, el sistema de vídeo puede bajar su ratio de bits para adaptarse a esta situación. Este tipo de sistemas ejecutan concurrentemente varias aplicaciones (por ejemplo, el decodificador de vídeo y el decodificador de audio), y cada una de ellas puede estar operando con unos ciertos parámetros en cada instante (lo que los autores llaman un "nivel de adaptación"). Mediante monitorización de este tipo de sistemas se pueden encontrar las frecuencias con las que cada tarea cambia de un nivel de adaptación a otro. Por tanto, el proceso de cambiar de niveles de adaptación puede ser modelado como una cadena de Markov de la que se puede extraer la probabilidad de que cada aplicación se halle en un nivel dado de adaptación.

Por otro lado, cada aplicación se compone a su vez de una secuencia de tareas, que se representan con un grafo. La primera tarea del grafo se activa periódicamente, y hay un plazo "extremo-a-extremo" para que la aplicación completa finalice. Se asume que este plazo es inferior al periodo de la aplicación. Cada una de las tareas que componen la aplicación tiene un tiempo de ejecución variable que es modelado mediante una función de probabilidad. Las diferentes tareas se mapean de forma estática a una red de procesadores.

A partir del modelo anterior, los autores se proponen obtener las probabilidades de que las aplicaciones cumplan sus plazos, cuando el sistema lleva ejecutándose un tiempo suficiente. Para ello caracterizan el estado del sistema mediante un conjunto de vectores. Así, por ejemplo, un vector contiene las tareas que están listas y cuánto tiempo de ejecución ha consumido cada una, otro vector contiene las tareas que se hallan en ejecución y cuánto le queda a cada una para terminar, y un tercer vector contiene el nivel de adaptación en que se halla cada una de las aplicaciones y cuánto tiempo hace que comenzó la ejecución de cada una. Tal como están definidos los estados, resulta que la probabilidad de que pase de uno a otro no depende de su historia pasada, y por tanto es una cadena de Markov que puede ser resuelta. Aunque el espacio de estados es muy grande, los autores aseguran que el análisis puede realizarse en poco tiempo. De este análisis obtienen las probabilidades de que el sistema esté en un estado dado, y de éstas se pueden deducir las probabilidades de que las aplicaciones cumplan sus plazos.

En su tesis doctoral [Manolache, 2002; Manolache *et al.*, 2001] utiliza unas técnicas muy similares para encontrar las probabilidades de perder el plazo en un conjunto de tareas que van a ejecutarse en un sistema multiprocesador conectado por una red. Su modelo es muy completo, pues incluye el modelado del sistema físico (conjunto de procesadores y buses que los unen) y un sofisticado modelo de ejecución de las tareas, en forma de un conjunto de grafos de precedencia en los que cada nodo representa una tarea y cada arco una relación de precedencia. Cada grafo tiene asociado un plazo relativo (que podríamos considerar del tipo "extremo-a-extremo" o "*end-to-end*") y un periodo, de modo que no son las tareas las que son periódicas, sino el grafo, o dicho de otro modo, la primera tarea del grafo. Si cualquiera de las tareas que componen el grafo excede el plazo

asociado con el grafo, se entiende que el grafo completo ha perdido su plazo. En este caso todas las tareas que lo componen podrán ser retiradas del sistema por el planificador, según la política que se considere para las tareas retrasadas. El planificador se basa en prioridades (asignadas estáticamente o dinámicamente, por lo que es válido tanto para RM como para EDF), pero sin embargo se impone de nuevo la severa limitación de que las tareas no pueden ser desalojadas una vez que entran en ejecución. Es decir, se trata de una planificación sin expulsión (*non-preemptive*). A partir de toda esta información, el análisis debe producir las probabilidades de pérdida de plazo de cada grafo (no de cada tarea, puesto que las tareas no tienen plazos).

Al igual que en [Kalavade y Moghê, 1998], el método de análisis se centra en encontrar el proceso estocástico subyacente y las probabilidades de cada uno de sus estados. El espacio de estados por el que se mueve el sistema se compone de n-tuplas de tareas. Cada posible estado del sistema indica qué tareas tiene en ejecución en un instante dado. Por ejemplo, un estado $\{\emptyset\}$ indica que no hay ninguna tarea en ejecución, mientras que un estado $\{\tau_1, \tau_2\}$ indica que una instancia de la tarea τ_1 y otra de τ_2 se hallan en ejecución. Para que el número de posibles estados sea finito, se asume que tan pronto como una tarea (o mejor dicho, el grafo de precedencia a que pertenece) pierde su plazo, es eliminada del sistema [5].

En cada instante, existe una probabilidad de que el sistema abandone el estado en que se halla y pase a un nuevo estado (bien porque una nueva tarea llega al sistema o bien porque una de las que estaba en ejecución ha finalizado o ha perdido su plazo y ha sido eliminada). El análisis para el caso monoprocesador se centra en calcular las probabilidades de pasar entre cada posible estado. Una vez estas probabilidades han sido obtenidas, el proceso estocástico queda completamente definido, pero en general no se trata de un proceso Markoviano (ya que las probabilidades de pasar de un estado a otro dependen del pasado del sistema). Para salvar esta dificultad, el autor aumenta el espacio de estados, introduciendo como parte del estado del sistema el instante en que entró en ejecución la tarea que está actualmente en ejecución. Se demuestra que en este caso el proceso embebido sí es Markoviano, y por tanto a partir de él se pueden encontrar las probabilidades a largo plazo de que el sistema se halle en cada uno de sus posibles estados, y de éstas las probabilidades de perder los plazos, para el caso monoprocesador.

Para el caso multiprocesador, al existir dependencias entre tareas situadas en diferentes procesadores, no se puede aplicar simplemente el análisis monoprocesador a cada uno de los procesadores. El intentar una técnica similar a la del caso monoprocesador, pero aumentando el espacio de estados para que cubra todos los procesadores, provoca una explosión de estados que vuelve el problema intra-

[5] En realidad el modelo permite que hasta b_i instancias de la misma tarea permanezcan en ejecución, siendo b_i un parámetro de la tarea i-ésima. Sin embargo, por simplicidad de la exposición, Manolache asume $b_i = 1$ para todas las tareas

table en la práctica. Este caso por tanto ya no es resuelto de forma exacta, sino que se introducen aproximaciones para hacerlo tratable. Así, cada tiempo de ejecución, que viene dado por una distribución de probabilidad genérica, se aproxima por una distribución exponencial o suma de exponenciales. El comportamiento global del sistema se modela mediante una red de Petri, que es posteriormente convertida en una cadena de Markov de tiempo continuo.

Tanto el método de Kalavade y Moghê [1998] como el de Manolache [2002] tienen tres restricciones importantes:

- Se limitan a mecanismos de planificación sin desalojo (*non-preemptive*), que no son demasiado utilizados en la práctica puesto que el desalojo permite un mejor aprovechamiento de los recursos.
- El modelo presupone que los plazos de los grafos (tareas *end-to-end*) han de ser iguales a sus periodos. Aunque posteriormente Manolache [2002] extiende el modelo para admitir plazos *inferiores* a los periodos, (sólo para el caso monoprocesador), el caso con plazos superiores a los periodos nunca es considerado.
- Aunque los autores no lo declaran explícitamente como una condición, en realidad el análisis sólo puede ser aplicado a sistemas en los que la utilización máxima sea inferior a 1. Esto se debe a que para encontrar una cadena de Markov en el proceso estocástico subyacente es necesario encontrar "puntos de renovación". Manolache [2002] encuentra estos puntos en el inicio de cada hiperperiodo, lo que supone que todas las tareas deben haber finalizado antes de que comience el hiperperiodo siguiente, para que de este modo todos los hiperperiodos comiencen de nuevo "desde cero". Esto sólo ocurrirá si la cantidad de trabajo generado en un hiperperiodo es siempre inferior a la longitud del hiperperiodo, es decir, si la utilización máxima es menor que 1.

2.3.4. Aplicando teoría de colas

Lehoczky desarrolla en dos publicaciones [Lehoczky, 1996, 1997] su teoría de "Colas de Tiempo-Real" (RTQT[6]), la cual pretende ser una ampliación de la teoría de colas clásica para incluir los requisitos temporales de los clientes en el modelo.

En la teoría de colas clásica, el estado del sistema usualmente se representa mediante un único escalar que es el número de clientes en el mismo. En el caso de querer aplicar este modelo a las tareas que se ejecutan en un procesador, el estado del sistema sería simplemente el número de tareas que se están ejecutando (o mejor dicho, que están listas y compitiendo por el procesador). A partir de este

[6] Real-Time Queuing Theory

modelo se pueden extraer conclusiones como el número promedio de tareas que se ejecutan, la utilización promedio del procesador, el tiempo promedio que un cliente debería estar esperando en la cola (o sea, el tiempo que deberá esperar hasta que el procesador quede libre y pueda comenzar su ejecución). Sin embargo este modelo no es adecuado para responder preguntas típicas del análisis de tiempo real, tales como la probabilidad de perder un plazo, dado que el modelo no incluye información alguna sobre plazos. Tampoco es posible extraer conclusiones acerca de la influencia de diferentes políticas de planificación o de asignación de prioridades.

Lehozcky propone extender el modelo de teoría de colas clásico, para incluir un parámetro adicional que pueda dar cuenta de los requisitos temporales de los clientes. En particular, el parámetro propuesto es el *plazo restante*[7], que se define como la diferencia entre el instante actual y el plazo absoluto de la tarea. De este modo, el estado del sistema en un instante ya no es simplemente el número de tareas listas, sino un vector que incluye para cada tarea cuál es su "plazo restante". El espacio de estados ya no tiene una sola dimensión, sino que de hecho el número de dimensiones no está acotado, ya que no lo está el número de tareas que pueden estar compitiendo por el procesador en un instante dado.

Como consecuencia, la resolución de este modelo se vuelve extraordinariamente compleja, y de hecho sólo se puede resolver de forma exacta para el caso en que las llegadas de las tareas sigan una distribución de Poisson y sus tiempos de ejecución sigan una distribución exponencial. En este caso, sabiendo el estado del sistema (es decir, el número de tareas de que consta y el plazo restante de cada una de ellas), es posible calcular las probabilidades de que en el instante siguiente esté en cualquier otro estado, gracias a que las probabilidades de que llegue un cliente nuevo, o de que uno de ellos finalice su ejecución, son fijas gracias a la propiedad "sin memoria" de las distribuciones exponenciales. De este modo, se encuentra que la evolución del sistema sigue un proceso de Markov y por tanto puede encontrarse su distribución de equilibrio. De este estado estacionario puede obtenerse el número esperado de clientes que perderán su plazo (pues serán los que tengan un "plazo pendiente" negativo).

En caso de que las llegadas no sigan una distribución de Poisson, o los tiempos de ejecución no sean exponenciales, el razonamiento anterior ya no es aplicable. Sin embargo, Lehoczky demuestra que, si la utilización promedio es muy próxima a 1 (lo que se conoce como la condición de *tráfico intenso*[8]), entonces es posible encontrar una solución aproximada.

El análisis para este caso se vuelve tremendamente complejo y abstracto. Básicamente se trata de realizar una transformación (reescalado) que convierte el modelo de colas en un movimiento Browniano reflejado, que se mueve en el espacio

[7] *lead-time*
[8] *heavy-traffic*

de las transformadas de Fourier. Esta transformación es válida cuando la utilización promedio es 1. Está demostrado que el comportamiento de la cola converge hacia este modelo a medida que su tráfico (utilización promedio) tiende a 1, con tal de que los tiempos entre llegadas sigan un proceso de renovación (no necesariamente Poisson) y con independencia de las distribuciones de los tiempos de ejecución.

En los trabajos de Lehoczky se comprueba por simulación si la solución obtenida bajo la hipótesis de tráfico intenso es aplicable a otros sistemas, y se encuentra que proporciona una aproximación bastante buena del ratio de tareas que perderán su plazo, *si bien no se encuentra un límite pesimista* (la cursiva es mía).

La ventaja del método es que permite su generalización hacia redes de colas [Lehoczky, 1997], y por tanto hacia sistemas multiprocesadores, y que permite analizar el impacto de diferentes políticas de planificación, ya que en el modelo la política de planificación es representada de forma abstracta como una operación entre transformadas de Fourier. Pero también presenta importantes inconvenientes:

- El modelo supone que los tiempos de ejecución de todas las tareas son variables aleatorias independientes e idénticamente distribuidas. Esta es otra restricción poco realista.

- Se asume que los instantes entre llegadas de las tareas siguen una distribución arbitraria. En el caso particular de una tarea periódica, esta distribución degeneraría en una distribución determinista. Pero en el caso de varias tareas periódicas, el tiempo entre activaciones de dos tareas cualesquiera no se puede caracterizar como una variable aleatoria, sin introducir un gran error. Por ejemplo, es típico que exista un instante crítico en que todas las tareas se activan a la vez. Esto quiere decir que la probabilidad de que el tiempo entre activaciones sea cero, no puede ser nula. Pero si damos una probabilidad no nula a este evento y tratamos el tiempo entre activaciones como una variable aleatoria, estamos permitiendo que un número arbitrario de activaciones pueda tener lugar a la vez (aunque sea con baja probabilidad). Es decir, estamos permitiendo en el modelo combinaciones de activaciones que no pueden producirse en la práctica.

- La hipótesis de tráfico intenso parece bastante arriesgada para sistemas de tiempo real. Si la utilización promedio es cercana a 1, la utilización máxima será probablemente muy superior a 1, dada la típica asimetría de los tiempos de ejecución de las tareas. Esto causará alta probabilidad de sobrecarga y por tanto de perder los plazos.

- El análisis presenta una gran complejidad teórica y no queda claro cómo se implementaría en una herramienta que pueda ser útil al analista.

2.3.5. Otros trabajos relacionados indirectamente

Mok y Chen [1997] proponen en su trabajo un modelo "multimarco", en el que cada tarea presenta diferentes tiempos de ejecución, pero se asume que estos tiempos aparecerán cíclicamente durante la ejecución del sistema. Por ejemplo, una tarea puede tener como tiempos de ejecución la secuencia $\{10, 20, 25\}$, lo que significa que en su primera activación su tiempo de ejecución sería 10, en la segunda sería 20 y en la tercera sería 25, y a partir de la cuarta los mismos tiempos se repetirían cíclicamente en ese orden. Este modelo ya representa una disminución de pesimismo con respecto al modelo clásico, que tomaría el tiempo peor de todos ellos (25) para todas las activaciones. Basándose en este nuevo modelo, Mok y Chen encuentran nuevos test de planificabilidad basados en el límite de utilización, menos pesimistas que los de Liu y Layland [1973]. Con todo, el tipo de análisis que proporciona este modelo es del tipo "si/no", es decir, el sistema cumplirá todos sus plazos, o no los cumplirá todos, pero no indica qué tareas no cumplirán sus plazos, ni la probabilidad de que tal cosa ocurra. Por otro lado, al asumir un patrón cíclico de tiempos de ejecución, su aplicabilidad queda restringida a los sistemas en los que aparezca este tipo de periodicidad en los datos.

En el trabajo de Zhou, Hu y Sha [1999] y en el de Hu y Zhou [2001] se parte también de un modelo en el que las tareas presentan tiempos de ejecución variables, modelados mediante su función de probabilidad, pero en lugar de buscarse el cálculo de la probabilidad de perder plazos, se define una nueva métrica "global": la probabilidad de que el sistema sea factible. Esta métrica se define como la probabilidad de que todas y cada una de sus tareas sean factibles, y esta a su vez se define como la probabilidad de que todas las instancias cumplan su plazo.

Los autores argumentan que disponer de una sola métrica global resulta más útil al analista a la hora de tomar decisiones de diseño para poder comparar diferentes diseños alternativos entre sí y decidir cuál es mejor.

En todo caso, la definición de "probabilidad de que el sistema sea factible" no parece demasiado clara. Los propios autores proponen al menos tres formas diferentes de interpretar intuitivamente la definición, dando lugar a tres formas diferentes de calcular esta probabilidad. Lamentablemente el resultado en cada caso sale también diferente. Por ejemplo, podría parecer a primera vista que esta probabilidad es el producto de las probabilidades de que cada una de las tareas cumpla su plazo. Sin embargo esto sólo es así si estas probabilidades son independientes y no lo son, incluso si se asume que las propias tareas son independientes. Otra interpretación consistiría en medir el número de instancias que cumplen su plazo, en un intervalo de tiempo, y dividirlo entre el número total de instancias generadas en ese intervalo, para después hacer tender a infinito el intervalo. Según los autores, este cálculo tampoco refleja correctamente el comportamiento probabilístico del sistema, aunque no explican por qué. Otra posibilidad es me-

dir la factibilidad a nivel de hiperperiodo[9]. Se dice que el hiperperiodo es factible si todas las tareas que se ejecutaron en él cumplen sus plazos. Si alguna falla, el hiperperiodo no es factible. Entonces podríamos medir la factibilidad del sistema como el número de hiperperiodos factibles dividido por el número total de hiperperiodos (a lo largo de un tiempo de observación que se hace tender a infinito). Esta interpretación tampoco es adecuada porque produce probabilidades de factibilidad muy bajas para el sistema (es raro el hiperperiodo en el que absolutamente todas las tareas cumplen su plazo).

Finalmente, la interpretación que los autores defienden se basa en el concepto de *ciclo de estado*[10] definido como el intervalo de tiempo que transcurre entre la activación de dos tareas sucesivas. En un ciclo de estado el conjunto de tareas que se está ejecutando en el sistema permanece fijo. El hiperperiodo se puede dividir en varios ciclos de estado, que se repetirán cíclicamente en los siguientes hiperperiodos. Los autores plantean medir la factibilidad a nivel de ciclo de estado. Un ciclo de estado se dice factible si todas las tareas que se están ejecutando en él cumplen sus plazos, y si alguna falla, se dirá no factible.

La probabilidad de que un ciclo de estado sea factible, es igual a la probabilidad de que todas y cada una de las tareas que tiene activas cumpla su plazo. Una vez obtenida la probabilidad de que cada ciclo de estado sea factible, la probabilidad de que el sistema completo sea factible se calcula como el promedio de las probabilidades de los ciclos de estado. Este cálculo es equivalente a dividir el número esperado de ciclos de estado factibles a lo largo de un hiperperiodo entre el número total de ciclos de estado en un hiperperiodo. Evidentemente el problema está en calcular la probabilidad de que un ciclo de estado sea factible. Los autores proponen plantear para cada ciclo de estado todas las combinaciones posibles de los tiempos de ejecución de las tareas que lo componen, y para cada una de esas combinaciones determinar si el ciclo es factible usando el análisis clásico, como por ejemplo el de la caracterización exacta del algoritmo RM de Lehoczky *et al.* [1989].

Aparte de que la métrica que se busca en este trabajo puede ser discutible, el modelo de partida presenta las siguientes limitaciones:

- No permite plazos mayores a los periodos
- No permite factores de utilización mayores de 1, puesto que asume que cada hiperperiodo es un punto de renovación.
- El método parece enfocado al planificador RM o DM, y no está claro cómo extender el método a EDF.

[9] El hiperperiodo es el mínimo común múltiplo de los periodos de todas las tareas
[10] *state cycle*

Kim *et al.* [2002] intenta resolver un problema diferente, pero utiliza un método de análisis con muchos puntos en común con el desarrollado en esta tesis. El problema que Kim *et al.* intentan resolver es el siguiente. Se tiene un conjunto de tareas periódicas, que serán planificadas bajo EDF y que presentan tiempos de ejecución muy variables, de los cuales se conoce su distribución estadística. Para dimensionar el sistema, en lugar de usar los peores tiempos de estas tareas como haría el análisis clásico, se usarán los tiempos promedio. Esto implica que, en promedio, las tareas cumplirán sus plazos, pero ocasionalmente se presentará una tarea que requiere más tiempo del previsto (lo que se denomina un *overrun*). El planificador debe decidir qué hacer con esta tarea. Las soluciones anteriores a este problema, básicamente trataban de aislar el efecto que esta tarea pueda tener sobre las demás, por ejemplo limitando su ejecución, ya sea eliminándola del sistema o posponiéndola. En cambio, en el trabajo de Kim *et al.* [2002] se plantea la posibilidad de que la tarea siga ejecutándose a su prioridad normal, o bien sea eliminada en base a una decisión aleatoria. El diseñador del sistema fijaría las probabilidades según las cuales la tarea sería eliminada o no. Cuando una tarea alcanza el tiempo que se le había supuesto, el planificador simplemente generará un número aleatorio que comparará con el suministrado por el diseñador, y en base a esa comparación eliminará la tarea o la dejará seguir ejecutándose[11] Según los valores que el diseñador dé a las probabilidades de eliminar tareas, cambiará la influencia que el *overrun* pueda tener sobre el resto del sistema.

Para evaluar el efecto de los parámetros de diseño sobre la fracción de plazos perdidos, los autores necesitan desarrollar un análisis estocástico del tiempo de respuesta. Este análisis se basa en considerar el tiempo de respuesta del primer trabajo de cada hiperperiodo como una cadena de Markov. El espacio de estados en que se mueve esta cadena es el de todos los posibles tiempos de respuesta. Ya que este espacio es infinito, la matriz de Markov que representa la cadena también será infinita. Resolver esta cadena (esto es, obtener la distribución estacionaria, o las probabilidades finales de cada posible tiempo de respuesta) implica hallar un "vector propio" de esta matriz infinita. Los autores no intentan resolver este problema y simplemente truncan la matriz para que tenga un tamaño finito y así poder aplicar métodos numéricos. Tampoco se demuestra si la solución estacionaria existirá siempre o bajo qué condiciones. Por otro lado, la obtención de los coeficientes de esta matriz es directa sólo cuando el planificador no admite expulsión. De no ser así, para poder encontrar la matriz de Markov, es necesario agrupar los trabajos según puedan expulsarse mutuamente o no, y analizar posteriormente la secuencia de "grupos" en lugar de la secuencia de trabajos. En ambos casos el método propuesto (basado en multiplicaciones de matrices dispersas) es muy ineficiente.

[11] En realidad, se propone que el planificador pueda hacer esto varias veces, en varios "puntos de control", y cada vez usar una probabilidad diferente

3. Modelo del sistema

> The justification of such a mathematical construct [model] is solely and precisely that it is expected to work.
>
> *(John Von Neumann)*

En este capítulo se presenta el modelo de sistema y se enuncia claramente el problema matemático que se busca resolver. En un primer momento, para que el análisis sea comprensible y tratable, el modelo está sometido a un conjunto de restricciones que restringen su aplicación en la práctica. Al final de este capítulo se señalan estas restricciones y se dan indicaciones de cómo podrían eliminarse para obtener un modelo de mayor aplicabilidad.

3.1. Modelo simplificado

El sistema se compone de un único procesador, al que van llegando **trabajos** en instantes conocidos de antemano, pero cuyos tiempos de ejecución son variables aleatorias de las que sólo se conoce su distribución estadística. Cada trabajo tiene asociada una prioridad P_j (que pudo ser asignada de forma fija durante la fase de diseño del sistema, o puede ser asignada de forma dinámica cuando el trabajo llega al sistema). La política de asignación de prioridades no nos incumbe. En el procesador se ejecuta un planificador, que asumimos no consume recursos, y que en cada instante garantiza que el trabajo que se está ejecutando es el de mayor prioridad de entre todos los trabajos listos. El planificador tiene capacidad de desalojo, esto es, si llega un trabajo con mayor prioridad que el que está actualmente en ejecución, éste último será suspendido para dejar ejecutarse al que acaba de llegar. Un trabajo suspendido será reanudado tan pronto como sea el trabajo de mayor prioridad de entre todos los trabajos listos.

Cada uno de los trabajos es una instancia de una **tarea**. Todos los trabajos correspondientes a una misma tarea tienen la misma distribución estadística de los tiempos de ejecución, el mismo plazo relativo (*deadline*) y en el caso de prioridades fijas, la misma prioridad P_i. Sólo se diferencian en el instante de activación, y en su prioridad para el caso de prioridades dinámicas. Los instantes de activación de los trabajos correspondientes a una misma tarea son deterministas, y están

separados por una distancia constante. Asumimos que los trabajos no acceden a recursos compartidos y por tanto no tienen secciones críticas en las que puedan quedar bloqueados.

3.1.1. Parámetros del sistema

Así pues, el sistema puede modelarse como un conjunto de tareas independientes periódicas $S = \{\tau_i\}$, con $i = 1, \ldots, N$, estando cada tarea definida por los siguientes parámetros $\tau_i = \{T_i, \Phi_i, D_i, \mathcal{C}_i\}$:

- **Periodo** T_i. Es el tiempo transcurrido entre dos sucesivas activaciones de la tarea,
- **Fase** u *offset* Φ_i. Es el instante de activación del primer trabajo de esa tarea,
- **Plazo** o *deadline* D_i. Es el plazo, relativo a su instante de activación, en el cual el trabajo debe haber finalizado su ejecución,
- **Tiempo de ejecución** $\mathcal{C}_i$. Es la cantidad de "recurso procesador" que solicita el trabajo, y equivale al tiempo que requeriría para su ejecución si comenzara a ejecutarse tan pronto como solicita el procesador y no fuera expulsado hasta que finalizara[1].

Mientras que los tres primeros parámetros son cantidades fijas y deterministas (conocidas de antemano), el último parámetro, el tiempo de ejecución $\mathcal{C}_i$ es una variable aleatoria. Para distinguir claramente a las variables aleatorias de las deterministas, usaremos un tipo de letra caligráfico (como $\mathcal{C}_i, \mathcal{R}_i$, etc.) para las primeras.

Por tanto, otro parámetro del sistema será la distribución estadística de los tiempos de ejecución $\mathcal{C}_i$ de cada una de las tareas. En lugar de requerir que estos tiempos sigan alguna distribución conocida (como la gaussiana, exponencial u otras), dejaremos que estas variables aleatorias sean de tipo genérico, con cualquier distribución. Por tanto, para completar la descripción del sistema, es necesario poseer la *función de probabilidad* de los tiempos de ejecución de las tareas. En general podríamos pensar que el tiempo de ejecución de una tarea puede tomar cualquier valor real, y en este sentido su función de probabilidad sería continua. En la práctica, en cambio, asumiremos una cierta *resolución* en la medida del tiempo, y tomando esta resolución como unidad temporal, los tiempos de ejecución sólo podrán tomar valores enteros, de modo que la función de probabilidad será discreta. Esta función discreta asigna una probabilidad a cada posible valor del

[1] No debe ser confundido con el tiempo de respuesta, que es el que transcurre desde que la tarea se activa hasta que finaliza su ejecución, pero tiene en cuenta el efecto de las restantes tareas del sistema que pueden retrasar su ejecución o expulsarla mientras se ejecuta.

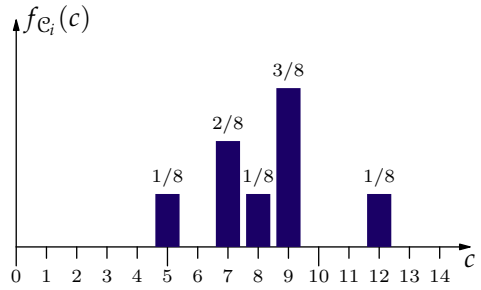


Figura 3.1.: Ejemplo de una función de probabilidad

tiempo de ejecución. Como ejemplo, véase la Figura 3.1, en la que se representa en abscisas los posibles valores del tiempo de ejecución y en ordenadas la probabilidad de cada uno. La altura de cada barra es proporcional a dicha probabilidad.

Denotaremos con $f_{\mathcal{C}_i}(\cdot)$ a la función de probabilidad de la variable aleatoria $\mathcal{C}_i$. Si denotamos por $\mathbb{P}\{x\}$ a la probabilidad del suceso x, tendremos que, por definición:

$$f_{\mathcal{C}_i}(c) \triangleq \mathbb{P}\{\mathcal{C}_i = c\} \tag{3.1}$$

En la Figura 3.1 puede verse, por ejemplo, que la probabilidad de que el tiempo de ejecución sea de 5 unidades, es de 1/8, la de que sea 7 unidades es 2/8, etc. Si la función asigna probabilidad cero a un tiempo de ejecución, es que ese tiempo no es posible. Así, en el ejemplo anterior no son posibles los tiempos 6,10,11 ni ninguno por encima de 12 o por debajo de 5. Aunque la función de probabilidad es en teoría infinita, en la práctica el tiempo de computación de una tarea τ_i estará comprendido entre dos valores finitos $C_i^{\min}$ y $C_i^{\max}$, por lo que la función de probabilidad $f_{\mathcal{C}_i}(\cdot)$ puede quedar completamente definida mediante un número finito de puntos. Una estructura obvia para almacenar este tipo de funciones sería el vector o *array* unidimensional, utilizando los posibles tiempos de ejecución como índices del *array* y siendo los valores almacenados en él las probabilidades de cada tiempo.

A partir de la función de probabilidad de una variable podemos conocer no sólo la probabilidad de que esa variable tome un valor dado, sino también la probabilidad de que tome valores por encima o por debajo de uno dado, mediante un simple sumatorio:

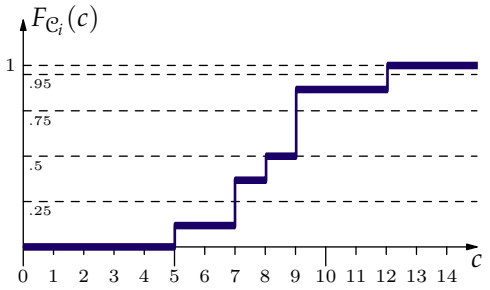


Figura 3.2.: Ejemplo de una función de distribución

$$\mathbb{P}\{\mathcal{C}_i \leq k\} = \sum_{j=-\infty}^{k} f_{\mathcal{C}_i}(j) = \sum_{j=0}^{k} f_{\mathcal{C}_j}(j) \tag{3.2}$$

$$\mathbb{P}\{\mathcal{C}_i > k\} = \sum_{j=k}^{\infty} f_{\mathcal{C}_i}(j) = \sum_{j=k}^{C_i^{\max}} f_{\mathcal{C}_j}(j) \tag{3.3}$$

A partir de la función de probabilidad $f_{\mathcal{X}}(\cdot)$ de una variable aleatoria $\mathcal{X}$, siempre es posible obtener la llamada **función de distribución acumulada**, que denotaremos por $F_{\mathcal{X}}(\cdot)$, y que se define como:

$$F_{\mathcal{X}}(x) \triangleq \mathbb{P}\{\mathcal{X} \leq x\} = \sum_{i=-\infty}^{x} f_{\mathcal{X}}(i) \tag{3.4}$$

Es decir, el valor de la función de distribución $F_{\mathcal{X}}(x)$ en un punto x es igual a la suma acumulada de la función de probabilidad $f_{\mathcal{X}}(\cdot)$ para todos los puntos menores o iguales que x. Por tanto, la función de distribución presentará una forma escalonada. Esta función, en general, tomará el valor 0 en $-\infty$, y el valor 1 en ∞. En nuestro caso particular, la función de distribución $F_{\mathcal{C}_i}(c)$ del tiempo de computación $\mathcal{C}_i$ tomará el valor cero para cualquier valor negativo de c, y el valor 1 para cualquier valor de c por encima de $C_i^{\max}$.

En la Figura 3.2 vemos un ejemplo de función de distribución. Se trata de la función de distribución correspondiente a la función de probabilidad de la figura 3.1. Observar que, por definición, la función de distribución se puede obtener a partir de la de probabilidad, y también, tomando las diferencias entre dos puntos sucesivos, es posible reconstruir la función de probabilidad a partir de la distribución. Es decir, ambas formas son equivalentes, por lo que cualquiera de las dos formas es aceptable como caracterización de la variable aleatoria $\mathcal{C}_i$ y por tanto cualquiera de las dos puede formar parte del modelo del sistema. En el futuro utilizaremos

una u otra según convenga, y abreviaremos "función de probabilidad" por PF[2] y "función de distribución acumulada" por CDF[3].

Denominaremos "**tiempo de respuesta**" de una tarea τ_i, y lo denotaremos por $\mathcal{R}_i$, al tiempo transcurrido desde el instante en que dicha tarea se activa hasta el instante en que finaliza su ejecución. Esta cantidad es una variable aleatoria. Si la tarea se ejecuta sola en el sistema (o si es la de mayor prioridad) su tiempo de respuesta coincidirá con su tiempo de ejecución, es decir, en este caso la PF de $\mathcal{R}_i$ coincidirá con la de $\mathcal{C}_i$. Sin embargo, en el caso general en que la tarea esté compitiendo con otras por el uso del procesador, deberá esperar hasta que el planificador le asigne este recurso, y sufrir las expulsiones que ocurran cada vez que una tarea con mayor prioridad sea activada, hasta que finalice su ejecución.

El concepto de tarea es en realidad una abstracción, puesto que lo que se ejecuta en el sistema son **trabajos**, que son las instancias de las tareas. Cada tarea τ_i da lugar a una infinita secuencia de trabajos $\Gamma_{i,j}$, todos ellos con la misma distribución de sus tiempos de ejecución, y con el mismo plazo relativo, pero con diferentes instantes de activación. El instante de activación del trabajo $\Gamma_{i,j}$, que denotaremos por $\lambda_{i,j}$, es determinista y viene dado por:

$$\lambda_{i,j} = \Phi_i + jT_i \tag{3.5}$$

En ocasiones necesitaremos referirnos a un trabajo genérico cualquiera, sin especificar la tarea a la que pertenece. En estos casos usaremos la notación Γ_j, es decir, un único subíndice en el trabajo. Esto significa simplemente el j-ésimo trabajo ejecutado desde el inicio del sistema. Para este propósito usaremos en general el subíndice j, y reservaremos el subíndice i para referirnos a tareas. Así, por ejemplo λ_j sería el instante de llegada del trabajo j-ésimo y $\mathcal{R}_j$ sería el tiempo de respuesta de ese trabajo. Sin embargo $\mathcal{R}_{i,j}$ sería el tiempo de respuesta de la j-ésima instancia de la tarea τ_i (es decir, del trabajo $\Gamma_{i,j}$). Finalmente $\mathcal{R}_i$ sería el tiempo de respuesta de la tarea τ_i, que se define como el promedio de los tiempos de respuesta de todos los trabajos que τ_i ha generado.

3.2. Enunciado del problema

> Dado un sistema S definido como en la sección anterior, obtener las funciones de probabilidad de los tiempos de respuesta de las tareas que lo componen.

[2] *Probability Function*. Algunos textos usan el término "Función de densidad" para referirse a este concepto, pero estrictamente hablando esto es incorrecto, ya que el concepto de función de densidad es aplicable sólo a variables aleatorias continuas. Para las discretas, el término correcto es "función de probabilidad" o "función de masa de probabilidad" (PMF)

[3] *Cumulative Distribution Function*

Este problema se representa gráficamente en la Figura 3.3, donde vemos que las "entradas" del análisis son los parámetros de cada una de las tareas del sistema, incluyendo entre estos parámetros las funciones de probabilidad de los tiempos de ejecución, y la "salida" del análisis es una función de probabilidad (del tiempo de respuesta) para cada tarea.

En la figura 3.4 se muestra un ejemplo del tipo de resultado que proporciona este análisis. En esta figura puede verse la función de probabilidad (PF) de una tarea, y su correspondiente función de distribución (CDF). En la primera podemos ver todos los posibles tiempos de respuesta de esa tarea, y la probabilidad de cada uno de ellos. En la segunda podemos leer directamente la probabilidad de que el tiempo de respuesta sea inferior a cualquier cantidad dada.

De cualquiera de las dos funciones se puede extraer directamente la probabilidad de que se cumpla el plazo (*PCP*), y la probabilidad de que se pierda el plazo (*PPP*). Si, por ejemplo, el plazo para esta tarea fuese de 69 unidades ($D_i = 69$), en la figura 3.5 se puede ver cómo se obtendrían los valores de *PCP* y *PPP* a partir de las funciones de probabilidad o de distribución. En el caso de que se use la función de probabilidad (Fig. 3.5(a)), el valor de *PCP* (probabilidad de cumplir el plazo) es simplemente la suma de todos los valores inferiores o iguales al plazo (barras de color azul en la gráfica). El valor de *PPP* (probabilidad de perder el plazo) es la suma de todos los valores por encima del plazo (barras rojas en la gráfica). Alternativamente puede obtenerse *PPP* como $1 - PCP$, y este segundo método puede ser el único viable si la función de probabilidad resulta tener una longitud prácticamente infinita (como veremos, este caso puede aparecer). Si en lugar de la función de probabilidad usamos la función de densidad (Fig. 3.5(b)), el valor de *PCP* se puede leer directamente sobre la curva, para la abscisa $D_i = 69$, y el valor de *PPP* es la distancia desde ese punto hasta 1.

Además de las probabilidades de cumplir o perder los plazos, las funciones de probabilidad y de distribución pueden proporcionar otros datos estadísticos que pueden ser de utilidad para el diseñador, como por ejemplo el tiempo medio de respuesta de una tarea, la desviación típica con respecto a ese tiempo medio, etc.

3.3. Limitaciones y ampliaciones del modelo

Como veremos en el siguiente capítulo, es posible encontrar una solución exacta al problema que se plantea en la sección anterior, gracias a las restricciones que se han impuesto al modelo del sistema. Sin embargo algunas de estas restricciones pueden limitar en exceso su aplicabilidad, por lo que sería interesante relajar algunas de ellas. Por desgracia, relajar cualquiera de ellas impedirá obtener una solución exacta del problema en un tiempo razonable (la complejidad del análisis se vuelve exponencial). No obstante será posible abordar algunas de ellas mediante simplificaciones apropiadas, que introducirán pesimismo en el análisis.

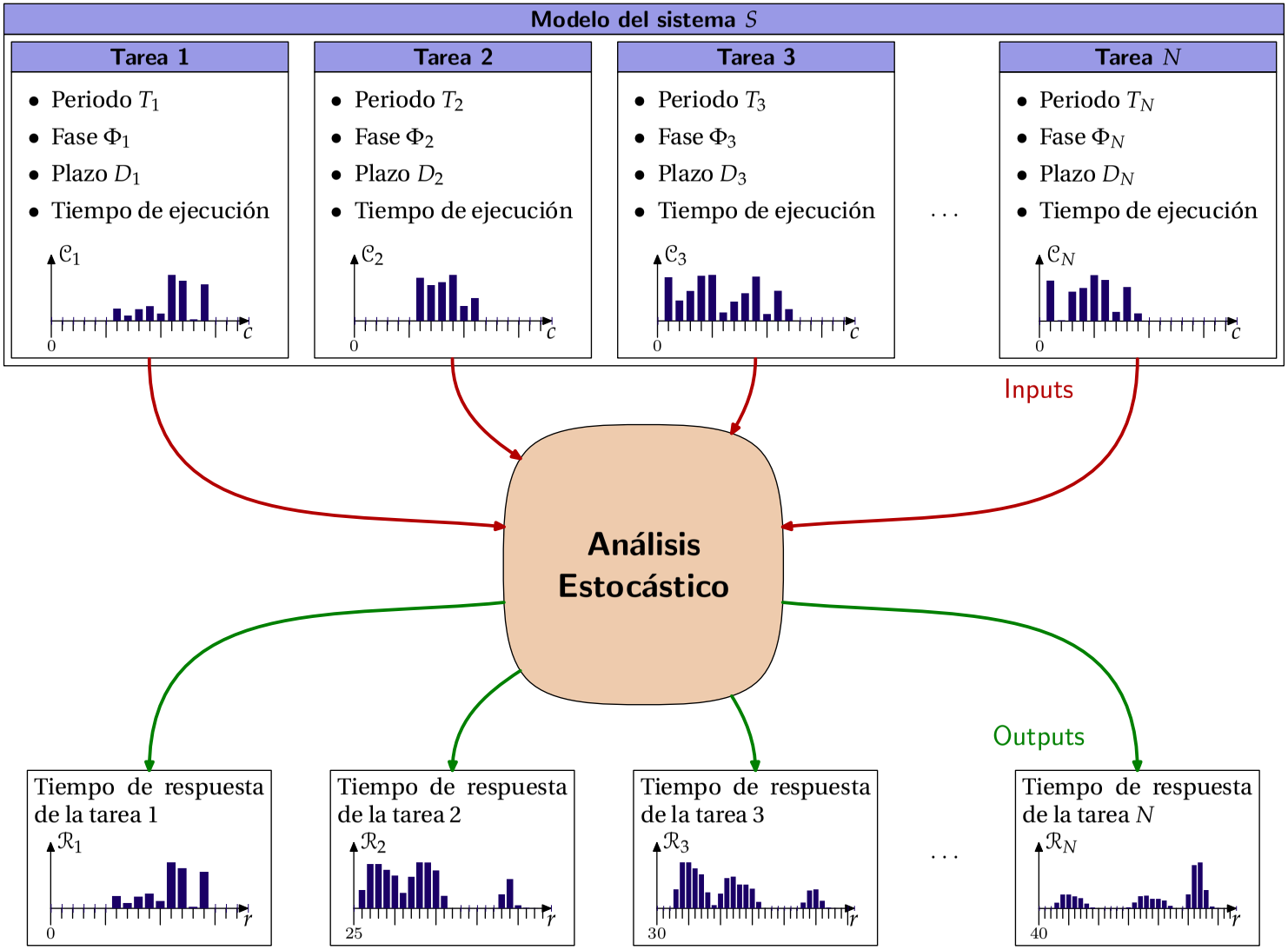
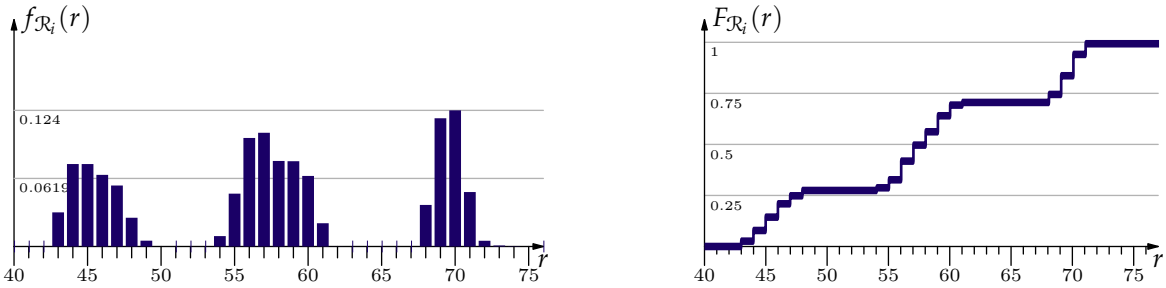


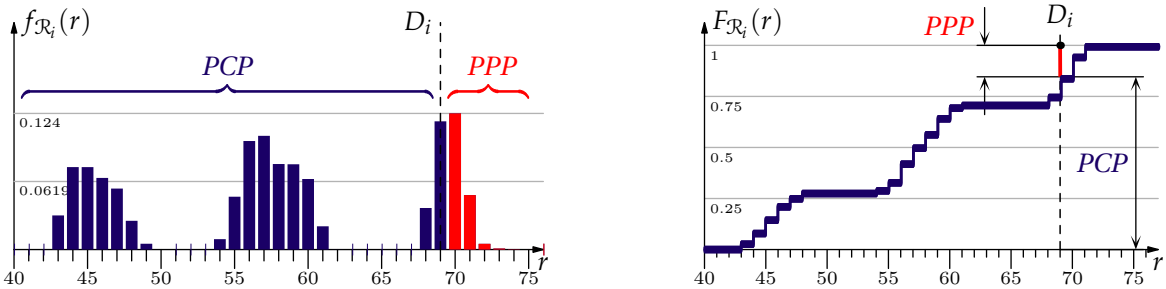
Figura 3.3.: Representación esquemática del problema a resolver en esta tesis



(a) Función de probabilidad (PF)

(b) Función de distribución (CDF)

Figura 3.4.: Ejemplo de funciones de probabilidad y de distribución de los tiempos de respuesta



(a) Sobre la función de probabilidad

(b) Sobre la función de distribución

Figura 3.5.: Ejemplo de obtención de las probabilidades de cumplir y perder un plazo

Las restricciones más importantes son las siguientes:

- **Suposición de independencia**. El modelo asume que las tareas no comparten recursos y que por tanto no pueden bloquearse debido a exclusiones mutuas. El modelo podría ampliarse suponiendo que las tareas tienen regiones críticas y utilizan algún protocolo de exclusión mutua en el acceso a las mismas, que evite el problema de la inversión de prioridades [Buttazzo, 1997; Burns y Wellings, 2001]. El instante exacto en que una tarea entra en su zona crítica, así como el tiempo que invertirá en ella, serán en general nuevas variables aleatorias. Sin embargo para que su análisis sea abordable será necesario tratar ambos como deterministas introduciendo un "peor caso". De este modo se pueden obtener probabilidades de pérdidas de plazo que

ya no son exactas, pero en todo caso son superiores a las reales. El modelo se extenderá para incluir estas consideraciones en el capítulo 5.

Existe otra suposición adicional de independencia, algo más sutil. Al definir el tiempo de ejecución de cada tarea mediante una función de probabilidad, estamos suponiendo implícitamente que no existen correlaciones estadísticas entre los tiempos de ejecución de las tareas, ni entre los tiempos de ejecución de dos instancias sucesivas de la misma tarea. Esto es, suponemos que el tiempo que requerirá una tarea al ejecutarse no depende del tiempo que ha requerido en la instancia previa, ni de los tiempos requeridos por otras tareas en el sistema. Al parecer, esta suposición no se cumple en la práctica Bernat *et al.* [2002]; Tia *et al.* [1995]. Si un trabajo tarda mucho en ejecutarse, hay mayores probabilidades de que el siguiente también tarde mucho (quizás porque toma como entrada datos que el otro ha producido, y el tiempo de ejecución es proporcional a los datos). También puede ocurrir el efecto contrario, si una tarea requiere mucho tiempo de ejecución en una de sus instancias, es posible que en la siguiente requiera menos. Esto puede ocurrir, por ejemplo, por efecto de un fallo de caché en la primera instancia, que sin embargo no se producirá ya en la segunda. Es decir, la correlación estadística se traduce en que las probabilidades reales de perder un plazo serán diferentes a las obtenidas suponiendo independencia. Y lo peor es que en algunas ocasiones serán superiores y en otras inferiores, por lo que el resultado del análisis no es válido ni siquiera como cota superior.

Relajar esta suposición de independencia estadística implicaría introducir en el modelo alguna forma de expresar las correlaciones entre los tiempos de ejecución de las diferentes tareas e instancias. Esto añadiría una gran complejidad al modelo y no se abordará en esta tesis.

- **Determinismo en los instantes de activación**. El modelo requiere que los instantes de llegada sean conocidos a priori. En la práctica esto excluye a las tareas de tipo esporádico o aperiódico. Además, no puede tener en cuenta el problema del *jitter*[4]3 .

  El problema del *jitter* también puede ser tratado en el análisis estocástico a costa de introducir pesimismo, como se verá en el capítulo 5. El problema de las tareas aperiódicas y esporádicas podrá ser resuelto introduciendo algún tipo de *servidor esporádico*, que limite la interferencia de este tipo de tareas sobre las otras. En el capítulo 5 se darán algunas indicaciones al respecto, si bien este tema no es tratado en profundidad en la presente tesis.

- **Falsas limitaciones**. Para simplificar la exposición y las demostraciones, el modelo tiene algunas suposiciones, que en realidad no son imprescindibles

[4] Se trata de una variación aleatoria alrededor del instante teórico de activación de los trabajos

para el análisis. Relajar estas limitaciones será trivial. Por ejemplo, el patrón de periodicidad de las tareas está limitado a las estrictamente periódicas, pero el análisis presentado en la tesis es igualmente aplicable a las esporádicamente periódicas, o a cualquier otro tipo de patrón que presente algún tipo de regularidad. Asimismo, asumiremos un planificador con desalojo, pero el caso de un planificador sin desalojo sería analizable también, de forma aún más simple. Estos aspectos también se tratarán en el capítulo 5 sobre extensiones al modelo.

4. Análisis

A reasonable probability is the only certainty.

(E.W. Howe)

4.1. Introducción y definiciones

El objetivo final del análisis es proporcionar las funciones de probabilidad de los tiempos de respuesta de cada tarea. Pero recordemos que el concepto de tarea es una abstracción, ya que lo que realmente se ejecuta en el sistema son los *trabajos* (instancias de las tareas). Cada trabajo puede tener un tiempo de respuesta con una función de probabilidad diferente. Considérese, por ejemplo, el caso en que todas las tareas tienen una fase cero. En el instante inicial llegarán muchos trabajos simultáneamente, por lo que el tiempo de respuesta de esta primera activación será diferente que el de otras activaciones sucesivas. La variable aleatoria "tiempo de respuesta" sigue una distribución diferente en cada activación de la tarea.

La distribución del tiempo de respuesta de una tarea podrá obtenerse promediando las distribuciones de los trabajos que la tarea engendra. Sin embargo el número de trabajos generados por una tarea es infinito, de modo que la pregunta es ¿puede obtenerse la distribución del tiempo de respuesta para una tarea sin necesidad de calcular las distribuciones de los infinitos trabajos?

Veremos en este capítulo que ello es posible, bajo ciertas condiciones, gracias a la periodicidad presente en los instantes de activación de los trabajos.

4.1.1. Factores que intervienen en el tiempo de respuesta de un trabajo

Supongamos por el momento que estamos interesados tan sólo en obtener la distribución del tiempo de respuesta de un trabajo concreto, es decir, de una instancia particular de una tarea, y no de la tarea en sí "como un todo".

El tiempo de respuesta de un trabajo depende no sólo de su propio tiempo de ejecución, sino también de los tiempos de ejecución de otros trabajos que puedan competir con él por el recurso procesador. Así, si cuando un trabajo llega el procesador está ocupado con otro trabajo de mayor prioridad, deberá esperar a que el

procesador le sea asignado, lo que incrementará su tiempo de respuesta. Además, una vez que el trabajo recibe el procesador, el planificador puede expulsarlo de nuevo si llega otro trabajo con mayor prioridad, en una planificación con desalojo (*preemption*). Esto también incrementará el tiempo de respuesta del trabajo.

Por consiguiente, en nuestro modelo, el tiempo de respuesta de un trabajo es la suma de tres factores:

1. La carga pendiente (o *backlog*) debido a trabajos de mayor prioridad que aún no ha sido servido cuando el trabajo se activa.

2. El tiempo de ejecución del propio trabajo

3. La suma de los tiempos de ejecución de los trabajos futuros que podrían expulsar al trabajo bajo consideración

Obsérvese que los tres factores son en realidad variables aleatorias, y sólo en el segundo de ellos la distribución estadística es conocida de antemano. Los restantes dependen del patrón de activaciones y de la asignación de prioridades. Este capítulo trata precisamente de encontrar un método para obtener las distribuciones estadísticas de los otros dos factores.

El primero de esos factores es un concepto de crucial importancia en esta tesis. Lo definiremos formalmente en el siguiente apartado. Veremos que la obtención de la distribución de la carga pendiente es la parte más compleja del análisis. Una vez esta distribución es conocida, la obtención de la distribución del tiempo de respuesta es bastante directa.

4.1.2. Carga del sistema

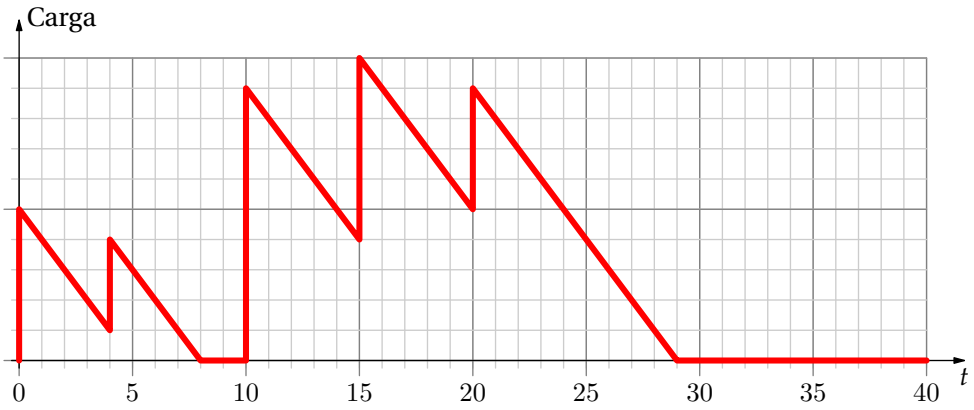
Definición 1. *La **carga del sistema** (o simplemente "carga"), en un instante dado t, es el tiempo de procesador que se necesitaría para finalizar todos los trabajos que se encuentran activos en el instante t. Se trata de una variable aleatoria que denotaremos por $\mathcal{W}(t)$.*

Cada vez que un nuevo trabajo llega al sistema, la carga se incrementa en una cantidad igual al tiempo de procesador demandado por dicho trabajo[1]. A medida que transcurre el tiempo, y mientras no lleguen trabajos nuevos, la carga se va decrementando (puesto que la tarea va recibiendo el tiempo de procesador que ha solicitado). Si eventualmente llega a cero, la carga se mantendrá en cero hasta la activación del siguiente trabajo.

[1] Este tiempo es una variable aleatoria, pero supondremos que el valor para esa instancia particular se "decide" en el instante en que el trabajo llega. Este es un artificio teórico que no resta generalidad al análisis.

Cuando la carga es cero, diremos que el procesador está **ocioso** o libre, y esto significa que en ese instante no hay ningún trabajo en ejecución. Si es distinto de cero, diremos que el procesador está **ocupado**.

La figura 4.1 muestra un ejemplo en el que cinco trabajos van llegando sucesivamente en los instantes 0, 4, 10, 15 y 20, y cada uno de ellos requiere respectivamente los tiempos de ejecución 5, 3, 9, 6 y 4. Se observa cómo la carga total pendiente sufre una discontinuidad (aumenta bruscamente) con la activación de cada nuevo trabajo, y va decayendo linealmente cuando no llegan trabajos nuevos. Observar que el sistema está ocupado desde el instante 0 al 7, ocioso desde el instante 7 al 10, ocupado de nuevo desde el 10 al 29, y finalmente ocioso otra vez a partir del instante 29. Naturalmente, la carga que cada trabajo añade es en realidad una cantidad aleatoria, por lo que en otro escenario la longitud de los periodos ocupados y ociosos sería diferente.



Llegan trabajos en los instantes 0, 4, 10, 15 y 20, cada uno con 5, 3, 9, 6 y 4 unidades de carga respectivamente

Figura 4.1.: Ejemplo de evolución de la carga en el sistema

Cuando un trabajo nuevo llega al sistema, habrá una cantidad de carga que retrasará su ejecución. Este retraso será igual al valor de la carga, si el trabajo que llega tiene prioridad inferior a todos los que se hallan en ejecución, pero será menor en caso contrario, ya que habrá que descontar la carga debida a trabajos de menor prioridad.

Por tanto hay dos tipos de trabajo en el sistema, según el retraso que experimenten sea igual a la carga del sistema, o sea menor. Esta clasificación de los trabajos en dos tipos resultará una pieza clave del análisis, por lo que la definiremos formalmente.

4.1.3. Trabajos basales y no-basales

Definición 2. *Diremos de un trabajo que es **trabajo basal** si, cuando llega al sistema, no hay en ejecución ningún otro trabajo con prioridad inferior a la suya. En ocasiones también nos referiremos a él como "trabajo de prioridad mínima".*

A la inversa, si cuando un trabajo llega, existe algún otro trabajo en ejecución con una prioridad menor, diremos que el trabajo es "no-basal" o de prioridad no-mínima.

Nuestro análisis requerirá que la asignación de prioridades sea tal que siempre se pueda determinar de antemano (fuera de línea) si un trabajo será basal o no. Se demostrará que las políticas de prioridades fijas cumplen esta propiedad, y que también la cumple EDF. Nuestro análisis no puede aplicarse a sistemas en los que la asignación de prioridades es tal que no permite esta clasificación a priori[2].

4.1.4. Hiperperiodo

Una vez que todas las tareas se han activado al menos una vez (es decir, una vez que haya transcurrido una cantidad de tiempo igual al máximo de las fases u *offsets* de todas las tareas), se producirá un patrón repetitivo en las activaciones de trabajos a partir de ese instante.

Esta idea es fundamental, ya que si existe un patrón repetitivo en las activaciones, cabe esperar algún tipo de comportamiento repetitivo del sistema. Incluso aunque los tiempos de ejecución sean aleatorios, cabe esperar que el comportamiento estocástico del sistema se haga repetitivo, es decir, se adapte siempre a unas mismas distribuciones estadísticas.

Definición 3. *El **hiperperiodo** es un fragmento de tiempo que recoge un conjunto de activaciones de tareas, tal que la secuencia relativa de estas activaciones se repetirá idénticamente en el futuro.*

La longitud del hiperperiodo, que denotaremos por T, es el mínimo común múltiplo de los periodos de todas las tareas.

$$T = \operatorname*{mcm}_i T_i \tag{4.1}$$

En la figura 4.2 puede verse un ejemplo con tres tareas, cuyas fases son 4, 7 y 11, y cuyos periodos son 6, 8 y 12. Para cada tarea se representa un eje de tiempos en el que una flecha vertical indica los instantes de activación de los trabajos. De acuerdo con la fórmula anterior, la longitud del hiperperiodo ha de ser 24 (mínimo común múltiplo de 6, 8 y 12). Efectivamente puede verse como, una vez que

[2] Por ejemplo, una mecanismo que asigne aleatoriamente la prioridad de un trabajo cada vez que llega uno, en tiempo de ejecución, sin atenerse a ningún patrón regular

todas las tareas se han activado al menos una vez (es decir, a partir del instante $t = 11$) existe un patrón de activaciones de longitud 24 que se repite una y otra vez. De hecho, a partir del instante 11 cualquier instante puede tomarse como inicio de un hiperperiodo, que se repetirá 24 unidades de tiempo después.

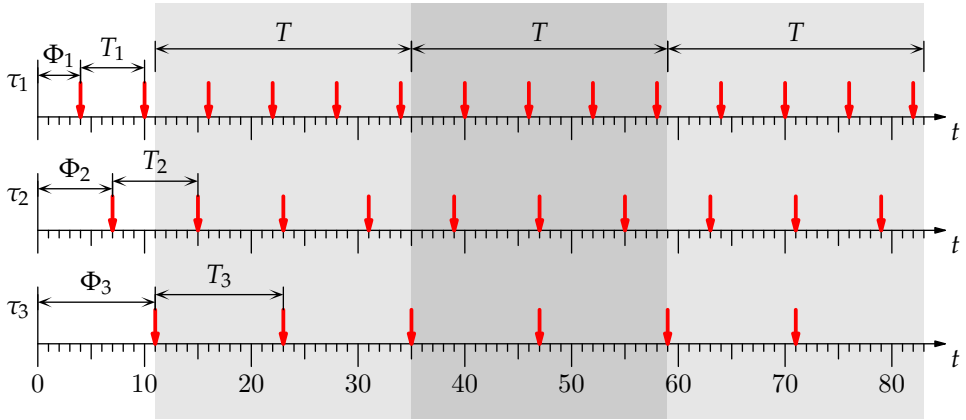


Figura 4.2.: Ilustración del concepto de hiperperiodo

El concepto de hiperperiodo proporciona un punto de partida para el análisis estocástico del sistema, que evita el tener que calcular la distribución del tiempo de respuesta de infinitos trabajos. En efecto, aunque la secuencia de trabajos sea infinita, el número de ellos dentro de un hiperperiodo es finito. Si en todos los hiperperiodos el comportamiento estocástico del sistema fuese el mismo, en el sentido de que las distribuciones de los tiempos de respuesta se repiten idénticas a las de los trabajos del hiperperiodo anterior, bastaría con calcular estas distribuciones para un hiperperiodo. A partir de ellas, promediando, se obtendría la distribución del tiempo de respuesta de las tareas.

Sin embargo, para que efectivamente el comportamiento estocástico del sistema sea repetitivo, no basta con que los instantes de activación lo sean. Además, la carga al inicio de cada hiperperiodo debe ser también la misma. De lo contrario, cada hiperperiodo comenzaría con diferentes condiciones iniciales, lo que afectaría a los tiempos de respuesta de los trabajos que lleguen en él.

En nuestro caso la carga es una variable aleatoria, por lo que no podemos exigir que sea la misma al inicio de cada hiperperiodo. En lugar de ello, exigiremos que su distribución estadística sea la misma. Es decir, si al comienzo de un hiperperiodo existe una probabilidad p_i de que la carga tenga el valor i, en el hiperperiodo siguiente exigiremos que esta probabilidad sea la misma. De este modo, todos los hiperperiodos tendrán las mismas "condiciones iniciales" (desde un punto de vista estocástico) y por tanto los trabajos presentarán las mismas distribuciones en sus tiempos de respuesta.

Cabe preguntarse bajo qué condiciones se cumple la exigencia anterior. Como

demostraremos a lo largo de este capítulo, hay casos en los que la distribución estadística de la carga del sistema es exactamente la misma al inicio de cada hiperperiodo y por tanto la condición anterior se cumple. En otros casos, la distribución no es exactamente la misma, pero las diferencias van tendiendo a cero a medida que el tiempo pasa, por lo que podemos asumir la existencia de un "régimen estacionario", que se alcanzará tras un suficiente número de hiperperiodos, y en el que la condición anterior también se cumple. Finalmente hay casos en los que la distribución de la carga cambia en cada hiperperiodo, y además las diferencias se acrecentan con el paso del tiempo. En este último caso la condición no se cumple, y el sistema no sería analizable con nuestro método. Por fortuna, veremos también que este último caso no presenta importancia práctica alguna, pues se trataría de sistemas irreales.

El parámetro clave que nos permitirá decidir en cuál de los tres casos anteriores nos encontramos es la "utilización", concepto que definiremos en el apartado siguiente.

4.1.5. Utilización

En el análisis clásico la utilización del sistema se define como la fracción de tiempo que el sistema permanecerá ocupado, en el peor caso. Alternativamente puede definirse como la fracción del hiperperiodo durante la cual el sistema permanece ocupado. Esta cantidad es muy sencilla de obtener, basta sumar el peor tiempo de ejecución de cada tarea dividido por su periodo, para todas las tareas del sistema. Si esta utilización máxima es mayor de uno, el sistema se considera no planificable, ya que una utilización mayor de uno significa que se está solicitando más tiempo de procesador del físicamente posible, esto es, durante un hiperperiodo se ha solicitado más trabajo que la duración del hiperperiodo mismo.

Sin embargo debemos tener en cuenta que para el cálculo de este factor de utilización se han usado los "peores casos" de los tiempos de ejecución. En la práctica, como sabemos, estos peores tiempos pueden aparecer con muy baja probabilidad. La posibilidad de que todas las tareas vayan a requerir su peor tiempo es más baja aún. Por tanto la "sobrecarga" en el sistema se producirá sólo ocasionalmente en algún que otro hiperperiodo. Evidentemente, cuando se dé ese caso, alguna tarea perderá su plazo, pero en promedio, el sistema podría estar ejecutándose durante mucho tiempo sin que esta sobrecarga aparezca y sin que ninguna tarea pierda su plazo. Por tanto tiene sentido preguntarse por las probabilidades de perder los plazos también cuando la utilización pueda ser mayor de uno.

Al enfrentarnos al caso estocástico, resulta evidente que cada hiperperiodo puede tener un factor de utilización diferente, y de hecho la utilización resulta ser otra variable aleatoria. Como tal, podemos hablar de su valor máximo, mínimo y pro-

medio, lo que origina las siguientes definiciones:

Definición 4. *La **utilización máxima**, denotada por U^{max} se define mediante la fórmula:*

$$U^{max} = \sum_{i=1}^{N} \frac{C_i^{max}}{T_i} \tag{4.2}$$

siendo C_i^{max} el peor tiempo de ejecución de la tarea τ_i. La utilización máxima representa la fracción del hiperperiodo durante el cual el sistema estará ocupado, en el peor caso.

Definición 5. *La **utilización mínima**, denotada por U^{min} se define mediante la fórmula:*

$$U^{min} = \sum_{i=1}^{N} \frac{C_i^{min}}{T_i} \tag{4.3}$$

siendo C_i^{min} el mejor tiempo de ejecución de la tarea τ_i. La utilización mínima representa la fracción del hiperperiodo durante el cual el sistema estará ocupado, en el mejor caso.

Definición 6. *La **utilización promedio** (o utilización media), denotada por $\bar{U}$ se define mediante la fórmula:*

$$\bar{U} = \sum_{i=1}^{N} \frac{\bar{C}_i}{T_i} \tag{4.4}$$

siendo $\bar{C}_i$ el valor medio del tiempo de ejecución de la tarea τ_i. La utilización media representa la fracción del hiperperiodo durante el cual el sistema estará ocupado, por término medio.

Obsérvese que nuestra definición de *utilización máxima* coincide con la definición clásica del *factor de utilización total.*

4.1.6. Régimen estacionario

Diremos que el sistema ha alcanzado un régimen estacionario cuando la distribución estadística de la carga del sistema sea la misma en dos hiperperiodos sucesivos (y por tanto, también en todos los hiperperiodos siguientes).

Algunos sistemas pueden alcanzar el régimen estacionario muy pronto. En otros es imposible que pueda alcanzarse. En realidad resulta muy sencillo saber de antemano en qué caso estamos, observando únicamente los valores de los parámetros $U^{\max}$ y $\bar{U}$ anteriormente definidos. Más adelante se demostrará matemáticamente esta afirmación, pero considero interesante adelantar este resultado. En

lo que respecta a la posibilidad de alcanzar el régimen estacionario, hay tres tipos de sistema:

1. Sistemas con $U^{\max} \leq 1$. En estos sistemas el régimen estacionario se alcanza "casi inmediatamente"; para ser más precisos, al final del primer hiperperiodo en el que todas las tareas han llegado al menos una vez. Por ejemplo, si todas las tareas tuvieran fase cero, el régimen estacionario se alcanzaría ya al final del primer hiperperiodo. Otro ejemplo; si el lector observa de nuevo la figura 4.2, verá que en este caso el primer hiperperiodo finaliza en el instante $t = 35$. Si este sistema tuviera una $U^{\max} \leq 1$ (lo que depende en realidad de las distribuciones de los tiempos de ejecución de los trabajos, lo cual no se muestra en la figura), el régimen estacionario se alcanzaría en ese instante $t = 35$.

   Este tipo de sistemas son muy interesantes desde un punto de vista práctico, ya que en realidad todos los sistemas analizables con los métodos clásicos eran de este tipo.

2. Sistemas con $U^{\max} > 1$ y $\bar{U} \leq 1$. En estos sistemas, aunque desde el punto de vista teórico siempre hay una diferencia entre la función de probabilidad de la carga pendiente de un hiperperiodo y la del siguiente, ocurre que esta diferencia tiende a cero. El sistema "converge" hacia un régimen estacionario que se alcanza, en teoría, tras un número infinito de hiperperiodos. Desde un punto de vista práctico, sin embargo, la diferencia puede llegar a ser despreciable en pocos hiperperiodos.

   Este tipo de sistemas son interesantes desde un punto de vista práctico, ya que es habitual que la utilización máxima (la del caso peor) sea muy grande, lo que haría el sistema no analizable por métodos clásicos, pero en cambio la utilización promedio no sea tan grande, lo que haría que las probabilidades de perder plazos sean bajas.

3. Sistemas con $\bar{U} > 1$. En estos sistemas no se puede alcanzar el régimen estacionario y por tanto no pueden ser analizados.

   Obsérvese no obstante que este tipo de sistemas no es realista, ya que en promedio se está exigiendo al sistema más tiempo del que éste puede ofrecer. Por tanto el que este tipo de sistemas no sea analizable no resta en realidad utilidad al análisis.

Veamos un ejemplo de cada uno de estos sistemas. En el cuadro 4.1 se muestran los parámetros de tres sistemas diferentes. Todos ellos constan de tres tareas, cuyos periodos, plazos y fases son idénticos para los tres ejemplos, variando únicamente las distribuciones de los tiempos de ejecución de las tareas. Obsérvese

		Parámetros				Utilización		
Sistema	Tarea	Φ_i	T_i	D_i	$\mathcal{C}_i$	$U^{\min}$	$\bar{U}$	$U^{\max}$
S_1	τ_1	4	6	6	U[1,2]	0.3750	0.6042	0.8333
	τ_2	7	8	8	U[1,2]			
	τ_3	11	12	12	U[1,3]			
S_2	τ_1	4	6	6	U[2,3]	0.7500	0.9792	1.2083
	τ_2	7	8	8	U[2,3]			
	τ_3	11	12	12	U[2,4]			
S_3	τ_1	4	6	6	U[2,4]	0.7500	1.125	1.5000
	τ_2	7	8	8	U[2,4]			
	τ_3	11	12	12	U[2,4]			

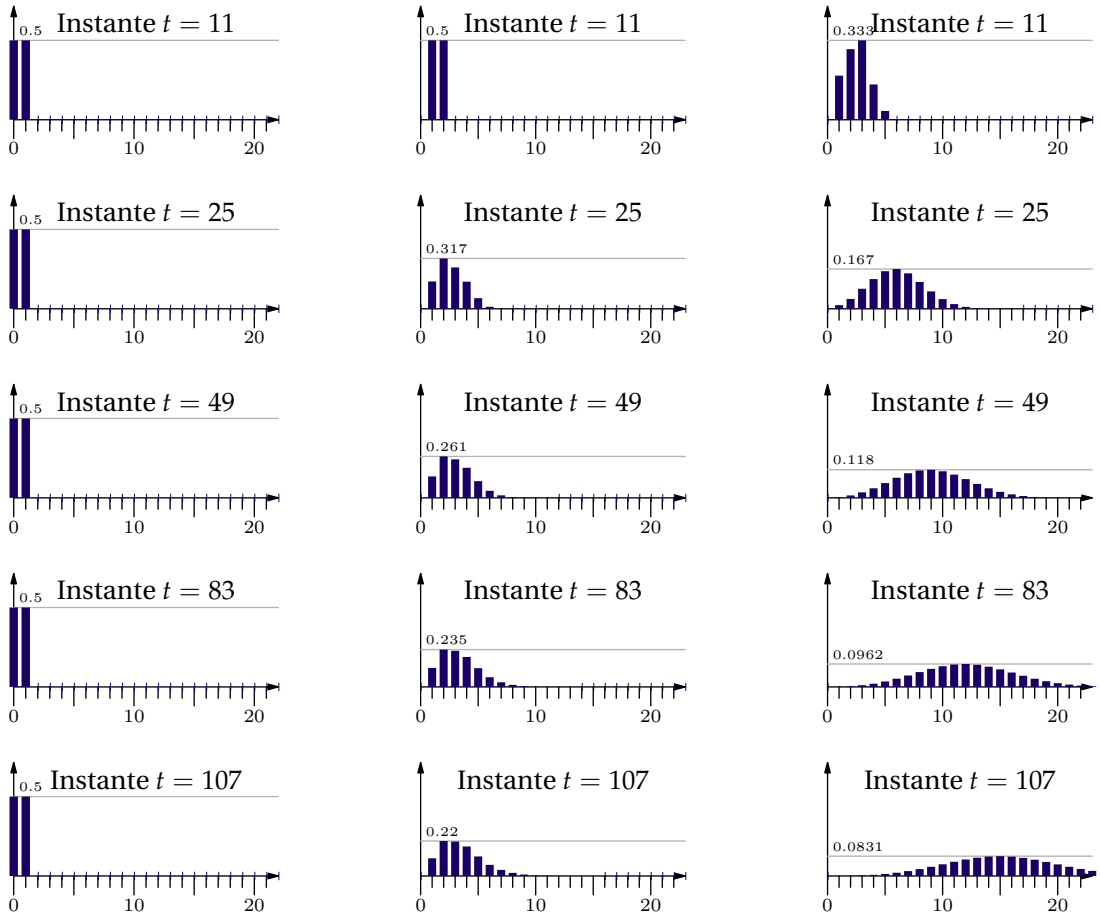
Cuadro 4.1.: Tres sistemas de ejemplo, con idénticos patrones de activación, pero diferentes factores de utilización

que los tres sistemas tienen por tanto el mismo patrón de activaciones de trabajos (que coincide además con el mostrado en la figura 4.2).

Los tiempos de ejecución de las tareas son variables aleatorias que siguen, en este ejemplo, una distribución discreta uniforme entre dos valores dados. Así, por ejemplo U[3,6] indica una distribución discreta uniforme entre los valores 3 y 6, lo que significa que esta variable aleatoria podría tomar los valores 3, 4, 5 ó 6, todos ellos con idéntica probabilidad (1/4 en este caso). La tabla recoge además los valores que tienen $U^{\max}$, $U^{\min}$ y $\bar{U}$ para este sistema, calculados de acuerdo con las fórmulas (4.2), (4.3) y (4.4), respectivamente.

El valor de $U^{\max}$ para el sistema S_1 es inferior a la unidad, por tanto, de acuerdo con la anterior clasificación, este sistema alcanzará su régimen estacionario al final del primer hiperperiodo. En la figura 4.3(a) se muestra la función de probabilidad de la carga total pendiente observada en diferentes instantes de tiempo, separados entre sí 24 unidades (que es la longitud del hiperperiodo). El primer instante mostrado corresponde a $t = 11$, que es el inicio del hiperperiodo (instante en que todas las tareas han sido activadas al menos una vez, véase la figura 4.2). Vemos como la función de probabilidad obtenida es exactamente la misma al principio de cada nuevo hiperperiodo.

El sistema S_2, en cambio, tiene un valor de $U^{\max}$ que excede la unidad. Pero, ya que la utilización promedio $\bar{U}$ es inferior a 1, de acuerdo con lo antes dicho este sistema también tendrá un régimen permanente que se alcanzará tras un número suficiente de hiperperiodos. En la figura 4.3(b) se muestra la función de probabilidad de la carga total pendiente observada en diferentes instantes separados 24 unidades. Vemos como en este caso las funciones de probabilidad no son idénticas, pero se parecen cada vez más a medida que pasan los hiperperiodos. Demos-



(a) Sistema S_1 (utilización máxima menor que la unidad)

(b) Sistema S_2 (utilización promedio menor que la unidad)

(c) Sistema S_3 (utilización promedio mayor que la unidad)

Figura 4.3.: Evolución de la distribución de la carga pendiente en los sistemas de ejemplo

traremos que en este caso existe una función límite hacia la que converge la PF de la carga pendiente, y proporcionaremos un método para obtener esta función.

Finalmente, el sistema S_3, al tener una $\bar{U}$ mayor que uno, carecerá de régimen permanente y por tanto no será analizable. La figura 4.3(c) muestra lo que ocurre en este caso con la función de probabilidad. Como vemos, esta función se va "estirando" y "aplastando", a la vez que se "aleja" del origen. Esto indica que la carga pendiente que se va acumulando al final de cada hiperperiodo puede crecer sin límite y que cualquier valor de la carga, por alto que sea, tiene una alta probabilidad de ser superado si se espera lo suficiente.

4.2. Visión general del método

En la sección anterior se ha definido el concepto de "carga del sistema" y se ha relacionado con el concepto de "hiperperiodo" para mostrar cómo el comportamiento estocástico de esta cantidad se "estabiliza" cuando la utilización promedio del sistema es inferior a uno. Aunque no se han proporcionado demostraciones de estas afirmaciones (se proporcionarán más adelante), se ha hecho hincapié mediante ejemplos en estos conceptos, porque resultan ser cruciales para el análisis estocástico propuesto.

En efecto, si somos capaces a determinar de alguna forma cuál será la distribución estadística de la carga al inicio de un hiperperiodo, a partir de ella será posible hallar la distribución estadística de la carga en cualquier otro instante dentro del hiperperiodo, mediante una sencilla técnica que desarrollaremos para esto. Esto nos permitirá encontrar la carga existente en el instante que un trabajo llega. Esta carga causará un retraso en la ejecución de dicho trabajo. La magnitud del retraso será igual al valor de la carga, si se trata de un *trabajo basal* (como se ha definido en la página 54). Si se trata de un *trabajo no-basal*, habrá que "descontar" de la carga aquella debida a trabajos de menor prioridad, ya que estos no causarán retraso al trabajo bajo consideración. Desarrollaremos un método para encontrar la distribución estadística de esta carga parcial.

Una vez tenemos la distribución estadística del retraso inicial (que es en realidad una variable aleatoria), el tiempo de respuesta del trabajo será como mínimo igual a la suma de este retraso más su propio tiempo de ejecución. Para obtener la distribución de la suma de dos variables aleatorias debe realizarse la *convolución* entre sus funciones de probabilidad. Más adelante volveremos sobre esto. El resultado de esta convolución sería directamente la función de probabilidad del tiempo de respuesta buscada, si el planificador utilizado no permitiera desalojo. En caso de que el planificador sí utilice desalojo, esta función de probabilidad obtenida por convolución debe "refinarse" para tomar en consideración el efecto de las expulsiones que el trabajo podría sufrir si llegan otros de mayor prioridad durante su ejecución.

Podemos pues resumir el proceso necesario para obtener la distribución del tiempo de respuesta de un trabajo en los siguientes pasos:

1. Obtener la distribución estadística de la carga del sistema, observada al principio del hiperperiodo de régimen estacionario.
2. A partir de la anterior, obtener la distribución estadística del retraso que experimentará el trabajo. Esto es muy sencillo para los trabajos basales y algo más complejo para los no-basales.
3. Convolucionar la distribución anterior con la del tiempo de ejecución del trabajo, para obtener una función de probabilidad que será la del tiempo de respuesta para los sistemas sin desalojo.
4. Para planificación con desalojo, la función obtenida en el paso anterior debe sufrir modificaciones que tengan en cuenta el efecto de las expulsiones.

Cada uno de los pasos anteriores requiere técnicas de análisis diferentes, que iremos introduciendo a lo largo de este capítulo, junto con las demostraciones matemáticas oportunas. De momento nos limitaremos a enunciarlas:

1. Para obtener la distribución estacionaria de la carga, será necesario aplicar teoría de procesos estocásticos y cadenas de Markov (sección 4.5).
2. A partir de la anterior podremos encontrar la distribución del retraso que sufrirá cualquier trabajo. No obstante la técnica es diferente según el trabajo sea de tipo basal o no:
   a) Para los trabajos basales, el retraso coincide con la carga del sistema. La distribución de la carga en cualquier instante, conocida la carga inicial (que se ha obtenido en el paso 1), puede hallarse mediante un sencillo método que denominaremos "convolucionar y encoger" (sección 4.3).
   b) Para obtener la distribución del retraso que afecta a los trabajos no-basales, es necesario aplicar una técnica que llamaremos "retroceder y rehacer", que básicamente sirve para "descontar" el efecto de los trabajos de menor prioridad que el considerado (sección 4.7).
3. Para combinar la distribución del tiempo de ejecución del trabajo con la distribución del retraso sufrido, usaremos la clásica operación de convolución discreta (ver apéndice A).
4. Para tomar en cuenta el efecto de las expulsiones de trabajos que llegan después, diseñaremos método que denominaremos "partir, convolucionar y juntar" (sección 4.4)

Figura 4.4.: Visión general del método y pasos requeridos

Los cuatro pasos del proceso se resumen gráficamente en la figura 4.4. Esta figura representa un eje de tiempos sobre el que se marcan los instantes de activación de diferentes trabajos. El tiempo de ejecución de cada trabajo viene definido por una función de probabilidad que se muestra también en la figura. Se asume que la figura representa un instante en la vida del sistema en el cual éste ya ha alcanzado el régimen permanente. Toda la historia pasada del sistema se resume en la "carga pendiente" al inicio del hiperperiodo (instante t_0). El primer paso (1) es encontrar la distribución de esta carga pendiente. Seguidamente, usando esta distribución y la de todos los trabajos que van llegando, el análisis "avanzará" hasta el instante t en el que llega el trabajo cuyo tiempo de respuesta queremos obtener (2). Obtendremos así[3] la distribución del retraso que sufrirá el trabajo que llega en t. Combinando éste con el tiempo de ejecución del trabajo, se tiene una primera "estimación" de su tiempo de respuesta (3), que debe ser refinada posteriormente tomando en consideración todos los trabajos futuros y analizando la posibilidad de que alguno de ellos pueda causar la expulsión y por tanto afectar al tiempo de respuesta (4).

El proceso anterior se traduce en realidad en dos metodologías ligeramente diferentes según estemos trabajando con prioridades fijas o dinámicas.

- Si usamos una política de prioridades fijas como *rate-monotonic*, todos los trabajos correspondientes a una misma tarea tienen la misma prioridad. Si consideramos la tarea de menor prioridad del sistema, todos sus trabajos serán basales, por lo que en el paso 2 sólo hay que aplicar el método "convolucionar y encoger". En cambio, los restantes trabajos del sistema serán no-basales, por lo que en apariencia habría que usar para todos ellos el método "retroceder y rehacer". Veremos que, para los sistemas de prioridades fijas, esto no será posible, pues el método "retroceder y rehacer" exigiría retroceder infinitos hiperperiodos hacia atrás, hasta el arranque del sistema. Para los sistemas de prioridades fijas resulta mucho más conveniente el plantear un nuevo sistema, idéntico al original, pero eliminando la tarea de menor prioridad. En este nuevo sistema, tendremos una nueva tarea de menor prioridad, cuyos trabajos serán basales y que se podrán resolver de la misma forma. Esta técnica puede repetirse para cada tarea del sistema.

  En definitiva, al trabajar con prioridades fijas necesitaremos repetir el análisis completo (lo que incluye el análisis Markoviano del paso 1) tantas veces como niveles de prioridad (tareas) haya en el sistema. Nótese sin embargo que en cada uno de estos análisis el sistema considerado tiene menos tareas y menor factor de utilización, por lo que su análisis es cada vez menos costoso.

[3] El método será "convolucionar y encoger" para los trabajos basales, y "retroceder y rehacer" para los no-basales

- En cambio, si usamos prioridades dinámicas como EDF, como demostraremos, el método "retroceder y rehacer" necesitará retroceder como máximo un hiperperiodo, por lo que el análisis de los trabajos no-basales puede lograrse sin necesidad de replantear y resolver un nuevo sistema. Para EDF, por tanto, el análisis Markoviano del paso 1 sólo hay que hacerlo una vez. Esto es interesante, puesto que este paso es con creces el más computacionalmente costoso de todo el análisis.

4.3. Cálculo de la carga en un instante dado. Convolucionar y Encoger

La carga es una variable aleatoria, que cambia en el tiempo, como se indica en su notación $\mathcal{W}(t)$. Esto implica que para diferentes instantes t, la función de probabilidad, $f_{\mathcal{W}(t)}(\cdot)$, de la variable es diferente. Nos proponemos en esta sección idear un método para obtener esta función de probabilidad en cualquier instante, si es conocida la de otro instante cualquiera anterior.

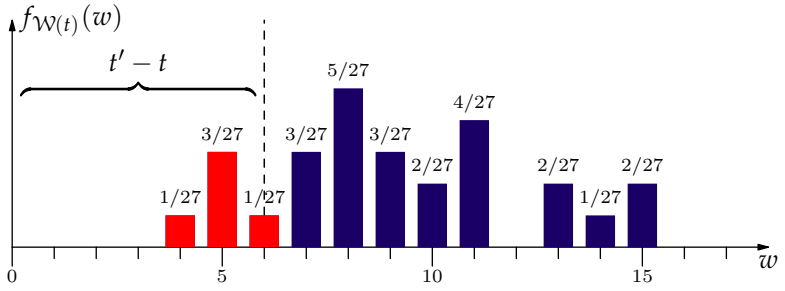
Supongamos que conocemos la distribución estadística de la carga en el instante t y queremos calcular cuál sería su distribución en otro instante t' posterior. Esto es sencillo si no llegan trabajos entre ambos instantes. Véase por ejemplo la figura 4.5(a), que representa una PF hipotética para la carga en el instante t. Consideremos que el otro instante t', dista 6 unidades de tiempo, y supongamos que no llegan más trabajos entre t y t'. Si la carga en t era de 6 unidades o menos, la carga en t' será de 0 unidades. La probabilidad de que esto ocurra es, en el ejemplo de la figura 4.5:

$$\mathbb{P}\{\mathcal{W}(t') = 0\} = \mathbb{P}\{\mathcal{W}(t) \leq 6\} = 1/27 + 3/27 + 1/27 = 5/27$$

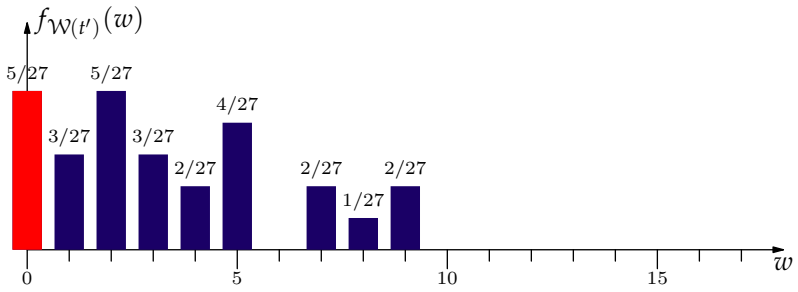
Si la carga en t era mayor que 6, la carga en t' será 6 unidades menor, ya que esta es la cantidad de tiempo transcurrida. Por tanto, podemos construir la PF de la carga en t' mediante un "desplazamiento" de 6 unidades a la izquierda de la PF en t, y una "acumulación" en el origen los valores de abscisa negativa tras el desplazamiento. El resultado de esta manipulación se muestra en la figura 4.5(b). Denominaremos a esta operación "encoger" la función de probabilidad en 6 unidades. Esta idea se formaliza en la siguiente proposición.

Proposición 1. *Si en el intervalo de tiempo $[t, t']$ no llega ningún trabajo, entonces*

$$f_{\mathcal{W}(t')}(w) = \begin{cases} \sum_{i=0}^{t'-t} f_{\mathcal{W}(t)}(i) & \text{para } w = 0 \\ f_{\mathcal{W}(t)}(w + t' - t) & \text{para } w > 0 \end{cases} \tag{4.5}$$



(a) Carga en el instante t



(b) Carga en el instante t' (6 unidades de tiempo después)

Figura 4.5.: Evolución de la función de probabilidad de la carga entre dos instantes cuando no llegan trabajos

Demostración. Puesto que no se añade al sistema carga alguna entre los instantes t y t', la carga en t' ha de ser igual a la que había en t, menos el tiempo transcurrido $(t'-t)$. Sin embargo, la carga no puede ser negativa, de modo que si la carga en t era menor que $t'-t$, entonces la carga en t' será cero. A partir de estas consideraciones la demostración de la proposición es inmediata. □

Veamos ahora el caso en que la carga es incrementada por la activación de un nuevo trabajo en el instante λ. La proposición siguiente relaciona la PF de la carga existente justo después del instante λ con la PF justo antes de dicho instante.

Proposición 2. *Denotemos por $\mathcal{W}(\lambda^-)$ a la carga existente en el instante $(\lambda-\epsilon)$, cuando $\epsilon \to 0$, y por $\mathcal{W}(\lambda^+)$ a la carga existente en el instante $(\lambda+\epsilon)$ cuando $\epsilon \to 0$. Si en el instante λ llega un único trabajo Γ, cuyo tiempo de ejecución, $\mathcal{C}$, sigue la función de probabilidad $f_{\mathcal{C}}(\cdot)$, entonces:*

$$f_{\mathcal{W}(\lambda^+)}(w) = \left(f_{\mathcal{W}(\lambda^-)} \otimes f_{\mathcal{C}}\right)(w) \tag{4.6}$$

donde $\otimes$ es el operador de convolución discreta (véase el apéndice A).

Demostración. La carga en el instante λ sufre una discontinuidad, puesto que es incrementada por el trabajo que llega, en una cantidad igual al tiempo de ejecución de este trabajo, $\mathcal{C}$. Ya que tanto la carga como el tiempo de ejecución del nuevo trabajo son ambas variables aleatorias, la PF de su suma se ha de obtener mediante la convolución de sus respectivas PFs, que es lo que se expresa en la ecuación (4.6). □

4.3.1. Algoritmo: convolucionar y encoger

Para simplificar el enunciado del algoritmo, hablaremos de instantes de activación en general, sin relacionarlos con las tareas que los originan. Así, en lugar de usar la notación $\lambda_{i,j}$ para denotar el instante en que se produce la j-ésima activación de la i-ésima tarea, usaremos simplemente λ_j para referirnos al j-ésimo instante en que llega algún trabajo. Es decir, $\{\lambda_j\}$ es la secuencia de instantes de activaciones, ordenada en el tiempo de modo que $\lambda_i \leq \lambda_j$ para todo $i < j$. Observar que puede haber varios λ_j seguidos con el mismo valor, si varios trabajos llegan en el mismo instante. Este enfoque, además de simplificar la notación, confiere mayor generalidad al algoritmo, ya que no lo restringe a sistemas de tareas periódicas, sino que sirve para trabajos con patrones de activación más complejos.

Supongamos que queremos conocer la función de probabilidad, f, de la carga en el instante λ_j^- (justo antes de λ_j) para un j arbitrario. El algoritmo a seguir será el siguiente:

1: $f \leftarrow$ PF de la carga inicial (dato de entrada)
2: $k \leftarrow 0$
3: **mientras** $\lambda_k < \lambda_j$ **hacer**
4: Convolucionar: $f \leftarrow f \otimes f_{\mathcal{C}_k}$
5: Encoger f en $(\lambda_{k+1} - \lambda_k)$ unidades
6: $k \leftarrow k + 1$
7: **fin mientras**

Observar que la función f, al ser discreta, se puede almacenar en realidad en forma de un array, $f[]$ tal que $f[x] = f(x)$. El array siempre tendrá un tamaño finito, puesto que existe una x a partir de la cual todos los valores de $f(x)$ son cero y no necesitan ser almacenados. El algoritmo es bastante eficiente en términos de uso de la memoria, ya que en cada iteración se reutiliza el espacio ocupado por f. Dicho de otro modo, no es necesario mantener en memoria los resultados de iteraciones anteriores. En cada iteración, f resume todo el pasado del sistema.

Observar asimismo que el algoritmo sigue siendo aplicable incluso si varios trabajos llegan en el mismo instante. Basta considerarlos de uno en uno. En este ca-

so, cada operación "encoger" deberá desplazar cero unidades a la izquierda la PF que ha recibido (puesto que la distancia entre λ_k y λ_{k+1} es cero). Por tanto, "encoger" no realiza en este caso ninguna operación, y por tanto el resultado del algoritmo será equivalente a convolucionar las PFs de todos los trabajos que llegan en ese instante, lo cual es perfectamente lógico, ya que todos sus tiempos de ejecución se suman. Además, el orden de las convoluciones no es importante (para los trabajos que llegan en el mismo instante), al ser la convolución una operación conmutativa.

Véase la página 81 para un ejemplo de cómo se aplica el algoritmo.

4.4. Cálculo del tiempo de respuesta. Partir, Convolucionar y Juntar

Supongamos que queremos obtener la función de probabilidad (PF) del tiempo de respuesta de un trabajo Γ_j. Mediante el algoritmo que acabamos de ver en la sección 4.3.1 podemos obtener la carga existente en el instante λ_j en que dicho trabajo llega. Esta carga causará un retraso en el trabajo bajo consideración. La magnitud del retraso depende de si el trabajo es de tipo basal o no-basal (ver definición en página 2). Ocupémonos de momento de los trabajos basales, y dejemos para más adelante (sección 4.7) el problema de los trabajos no-basales.

Un trabajo basal tiene menor prioridad que todos los que se están ejecutando cuando él llega. Por tanto antes de poder comenzar su ejecución deberá dejar que transcurra un tiempo igual a la carga existente en el momento de su activación. Por tanto, su tiempo de respuesta será como mínimo igual a la suma de la carga existente a su llegada más su propio tiempo de ejecución. De hecho, si no llegaran más trabajos después del instante λ_k, tendríamos que esta suma sería exactamente el tiempo de respuesta del trabajo Γ_k. En este caso, ya que ambas magnitudes son variables aleatorias, la distribución de este tiempo de respuesta se obtendría mediante una convolución discreta.

Por desgracia pueden llegar más trabajos después del instante λ_k, y algunos de ellos expulsarán al que estamos considerando, si éste aún está ejecutándose cuando el nuevo llega, y si el nuevo tiene mayor prioridad que el considerado. Si se produce una expulsión, el tiempo de respuesta aumentará.

De cara a clarificar el problema y el método para resolverlo, desarrollaremos un ejemplo simple. Una vez que las ideas clave han sido presentadas a través de este ejemplo, el método será formalizado en un teorema.

Trabajo	λ_i	P_i	$f_{\mathcal{C}_i}(\cdot)$
Γ_1	0	5	Discreta U[2,5]
Γ_2	3	10	Discreta U[1,2]
Γ_3	6	15	Discreta U[1,3]

Cuadro 4.2.: Sistema de ejemplo para el cálculo del tiempo de respuesta

4.4.1. Ejemplo preliminar

Considérese el sistema mostrado en el cuadro 4.2. Queremos obtener la PF del tiempo de respuesta para el trabajo Γ_1, que llega en el instante $\lambda_1 = 0$. Observar que no estamos planteando (aún) un modelo de tareas periódicas, sino una simple secuencia de trabajos que llegan una sola vez. Cada trabajo tiene una prioridad pre-asignada, sin que nos importe cuál ha sido la política de asignación. Para este ejemplo supondremos además, que no hay carga inicial. Por tanto, en el instante en que llega el trabajo Γ_1 la carga es cero con probabilidad 1.

Como una primera aproximación, nos planteamos obtener la función de probabilidad del tiempo de respuesta $\mathcal{R}_1$ bajo una hipótesis simplificadora: ningún trabajo podrá expulsar al que estamos considerando. Bajo esta hipótesis, el tiempo de respuesta será igual a la carga en el instante λ_1 más el tiempo de ejecución del trabajo Γ_1. Ya que ambas magnitudes son variables aleatorias, la función de probabilidad de su suma será igual a la convolución de sus funciones de probabilidad. En este ejemplo, la carga inicial es cero, es decir, tiene una función de probabilidad que vale 1 en cero y 0 en los restantes puntos. Al convolucionar esta función con $f_{\mathcal{C}_1}(\cdot)$, obtendremos de nuevo $f_{\mathcal{C}_1}(\cdot)$.

Ahora debemos revisar la hipótesis simplificadora ¿es cierto que ningún trabajo futuro podrá expulsarle? La respuesta depende de en qué instante llegue el próximo trabajo y de qué prioridad tenga. Si el siguiente trabajo llega 5 unidades de tiempo o más tarde, entonces efectivamente ya no podrá expulsarle, porque para entonces el trabajo ya habrá terminado (ya que su peor tiempo de ejecución es 5). Sin embargo, vemos que en este ejemplo el siguiente trabajo llega tan sólo 3 unidades de tiempo después. Por tanto existe una probabilidad de que Γ_1 aún esté ejecutándose.

Observar sin embargo un hecho interesante: la probabilidad de que el tiempo de respuesta sea 0, 1, 2, ó 3, puede obtenerse directamente de la función de probabilidad obtenida bajo la hipótesis simplificadora, ya que si los tiempos de respuesta tomaran estos valores, el trabajo Γ_1 finalizaría antes de darle oportunidad a Γ_2 de expulsarle. Por tanto los primeros puntos de la función obtenida bajo la hipótesis simplificadora son válidos. Sólo los puntos con abscisa 4 o superior no valen y deben ser recalculados.

Si lo hacemos (con el método que veremos enseguida), tendremos una segunda aproximación al tiempo de respuesta, mejor que la aproximación inicial puesto que ahora ya tiene en cuenta el efecto del trabajo Γ_2. Aunque no hemos tomado

en consideración el trabajo Γ_3, la segunda aproximación es válida para valores de la abscisa menores o iguales que 6, puesto que si el tiempo de respuesta tomara uno de estos valores, Γ_1 habría acabado antes de dar oportunidad a Γ_3 de que le expulse. Sólo los valores con abscisa mayor de 6 deberán ser recalculados para tomar en cuenta la interferencia de Γ_3.

Y así sucesivamente. Vemos que el método propuesto para obtener la PF del tiempo de respuesta $\mathcal{R}_1$ involucrará varios pasos. En el paso "cero", obtenemos la PF de $\mathcal{R}_1$ bajo la hipótesis de que no sufrirá expulsión alguna. En el paso uno, modificaremos esta PF para tener en cuenta el efecto del primer trabajo que podría sufrir, el causado por el trabajo Γ_1. En el paso dos lo modificaremos de nuevo para tener en cuenta el siguiente posible desalojo, etcétera. Denotaremos por $\mathcal{R}_j^{\langle k \rangle}$ al tiempo de respuesta del trabajo Γ_j, obtenido en el paso k-ésimo de nuestro método; es decir, el tiempo de respuesta que saldría al considerar tan sólo las k primeras expulsiones. Apliquemos este método a nuestro ejemplo.

Paso cero: En nuestro ejemplo, $f_{\mathcal{R}_1^{\langle 0 \rangle}}(\cdot)$ será igual a $f_{\mathcal{C}_1}(\cdot)$, ya que hemos supuesto que no había carga inicial. En la figura 4.6(a) se muestra la gráfica de esta función. Como hemos dicho, sus primeros puntos (marcados en la figura con una llave) nos dan parte de la PF de $\mathcal{R}_1$ buscada, ya que:

$$\mathbb{P}\{\mathcal{R}_1 = r\} = f_{\mathcal{R}_1^{\langle 0 \rangle}}(r) \qquad \text{para } r \leq \lambda_2 - \lambda_1$$

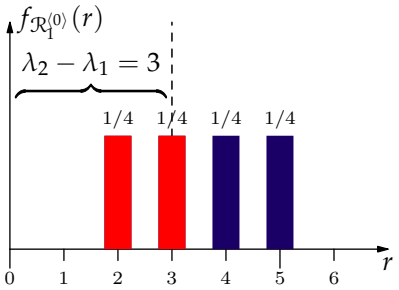
Podemos dividir la función $f_{\mathcal{R}_1^{\langle 0 \rangle}}(\cdot)$ en dos mitades, para quedarnos con esos puntos válidos. Para ello tomamos como punto de división la abscisa 3 (igual a $\lambda_2 - \lambda_1$). Como se muestra en la figura 4.6(b), la "mitad inferior" resultante se construye copiando los valores de la función $f_{\mathcal{R}_1^{\langle 0 \rangle}}(\cdot)$ hasta el 3, inclusive, y dejando a cero todos los restantes. Para la "mitad superior", el proceso es el contrario; se rellenan con ceros todos los valores hasta el 3 inclusive, y se copian de $f_{\mathcal{R}_1^{\langle 0 \rangle}}(\cdot)$ los restantes.

Esta operación de "partir en dos" una función, puede ser definida más formalmente con ayuda de la siguiente notación:

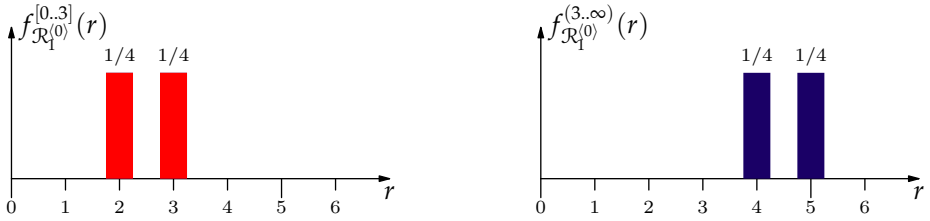
Definición 7. *Dada una función $f(\cdot)$ y un intervalo real I, definiremos la función $f^{\mathbf{I}}(\cdot)$ en la forma siguiente:*

$$f^{\mathbf{I}}(r) = \begin{cases} f(r) & si\ r \in \mathbf{I} \\ 0 & si\ no \end{cases} \tag{4.7}$$

Es decir, $f^{\mathbf{I}}(\cdot)$ copia los valores de $f(\cdot)$ dentro del intervalo $\mathbf{I}$, pero vale cero fuera de él. Usando esta notación, tras "partir" la función $f_{\mathcal{R}_1^{\langle 0 \rangle}}(\cdot)$ en la abscisa 3, la mitad inferior resultante (parte izquierda de la figura 4.6(b)) se denotará por



(a) Función de probabilidad de $\mathcal{R}_1^{\langle 0\rangle}$. Se señala la zona que es "aprovechable"



(b) Resultado de "partir" en la abscisa 3

Figura 4.6.: Cálculo de la PF del tiempo de respuesta. Paso cero.

$f^{[0..3]}_{\mathcal{R}_1^{\langle 0\rangle}}(\cdot)$, y la mitad superior (parte derecha de la misma figura) por $f^{(3..\infty)}_{\mathcal{R}_1^{\langle 0\rangle}}(\cdot)$. Observar que el intervalo es cerrado para la mitad inferior pero abierto para la mitad superior.

La mitad inferior se "guarda" como parte de la respuesta buscada, mientras que la mitad superior necesita más elaboración.

Paso uno: Consideremos ahora el cálculo de $\mathbb{P}\{\mathcal{R}_1 = r\}$, para $r > (\lambda_2 - \lambda_1) = 3$, por ejemplo para $r = 5$. En este caso, tenemos dos factores a tener en cuenta:

- Este tiempo de respuesta, al ser mayor de 3, debe incluir la interferencia debida al trabajo Γ_2.
- Pero por otro lado, este caso sólo puede darse si, ya de por sí, el tiempo de ejecución del trabajo Γ_1 (sin contar la interferencia) era mayor de 3, ya que si fuera menor, habría terminado antes de que Γ_2 llegara.

Por tanto, la probabilidad de que el tiempo de respuesta sea 5, es la probabilidad de una intersección de eventos: que el tiempo de ejecución, sin contar la interferencia, sea mayor de 3, y que el tiempo de ejecución más la interferencia sea igual a 5. Es decir:

$$\mathbb{P}\{\mathcal{R}_1 = 5\} = \mathbb{P}\{(\mathcal{C}_1 > 3) \wedge (\mathcal{C}_1 + \mathcal{C}_2 = 5)\}$$

Esta probabilidad puede obtenerse sumando las probabilidades de todos los posibles casos que se hallen en la intersección:

$$\mathbb{P}\{\mathcal{R}_1 = 5\} = \sum_{i=3}^{5} \mathbb{P}\{\mathcal{C}_1 = i\} \cdot \mathbb{P}\{\mathcal{C}_2 = 5 - i\}$$

Como vemos, el sumatorio anterior se parece a una convolución, sólo que incompleta. En realidad el resultado coincide con la convolución de $f_{\mathcal{C}_2}(\cdot)$ con la "mitad superior" $f_{\mathcal{R}_1^{\langle 0 \rangle}}^{(3..\infty)}(\cdot)$ que habíamos separado antes. Este razonamiento es válido para cualquier $r > (\lambda_2 - \lambda_1)$. Por tanto, podemos "refinar" la mitad superior simplemente convolucionándola con la PF del tiempo de ejecución del trabajo que la interrumpe. La figura 4.7(a) muestra esta operación.

Por otro lado, para los valores de r menores ya habíamos visto que las probabilidades venían dadas por la "mitad inferior" $f_{\mathcal{R}_1^{\langle 0 \rangle}}^{[0..3]}(\cdot)$. Por tanto, juntando ambos resultados podremos construir la función de probabilidad del tiempo de respuesta, que toma en consideración el efecto de la primera expulsión. Esto es, obtendremos la función de probabilidad de $\mathcal{R}_1^{\langle 1 \rangle}$:

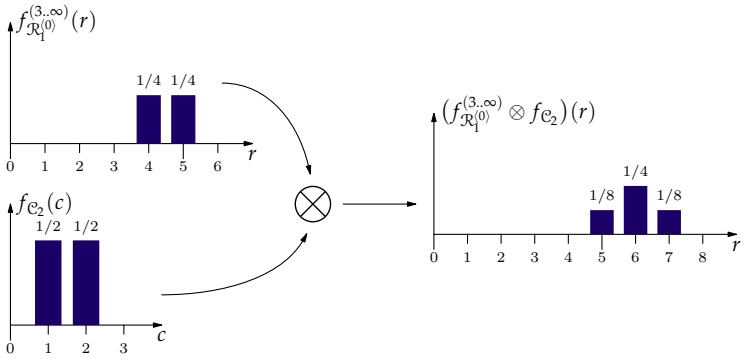
$$f_{\mathcal{R}_1^{\langle 1 \rangle}}(r) = f_{\mathcal{R}_1^{\langle 0 \rangle}}^{[0..3]}(r) + \left(f_{\mathcal{R}_1^{\langle 0 \rangle}}^{(3..\infty)} \otimes f_{\mathcal{C}_2}\right)(r)$$

Observar que la operación de "juntar" se expresa mediante una simple suma, pues las funciones son complementarias (donde una es distinta de cero, la otra es cero). La figura 4.7(b) muestra esta operación.

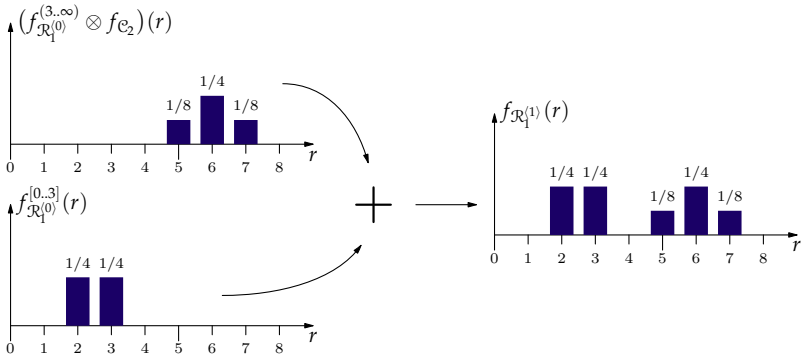
Paso dos: Ahora es necesario tomar en cuenta la interferencia debida al siguiente trabajo Γ_3. Para ello aplicaremos las mismas ideas desarrolladas en el paso uno. La función $f_{\mathcal{R}_1^{\langle 1 \rangle}}(\cdot)$ obtenida en el paso previo se "parte" en dos, siendo el punto de corte igual a 6 (que es $\lambda_3 - \lambda_1$). La mitad inferior se guarda, como parte de la función final buscada. La mitad superior se convoluciona con la función de probabilidad del tiempo $\mathcal{C}_3$, del siguiente trabajo. El resultado de esta convolución se "junta" con la mitad inferior previamente guardada y de este modo se obtiene la función $f_{\mathcal{R}_1^{\langle 2 \rangle}}(\cdot)$. Todo este proceso se ilustra gráficamente en la figura 4.8, donde las etapas "partir", "convolucionar" y "juntar" son claramente visibles.

Pasos siguientes: El proceso anterior se repetiría para cada uno de los trabajos futuros que vayan llegando. En nuestro ejemplo ya no llegan más, por lo que la función obtenida en el paso anterior es ya la buscada $f_{\mathcal{R}_1}(\cdot)$.

Observar que en cada paso se obtiene una función que es válida en un rango de puntos iniciales. Este rango se va extendiendo con cada nueva iteración. Puede darse el caso de que el rango cubra la función completa (esto es, fuera de este rango sea cero). Esto indicaría que cualquier trabajo futuro ya no podría causar



(a) Convolución de la "mitad superior" con la carga del siguiente trabajo



(b) Juntar el resultado con la "mitad inferior" antes guardada

Figura 4.7.: Cálculo de la PF del tiempo de respuesta. Paso uno.

interferencia, puesto que el trabajo ya habría finalizado su ejecución incluso en el peor caso posible.

Observar asimismo que, si el análisis no necesitara la función de probabilidad completa, sino que se conforma con saber la probabilidad de que el tiempo de respuesta esté por debajo de cierto valor (por ejemplo, la probabilidad de que se cumpla un cierto plazo), esta probabilidad puede obtenerse sumando los primeros puntos de la función. Si los puntos sumados están todos ellos dentro del rango de validez, la probabilidad así obtenida será exacta.

Las dos observaciones anteriores implican que, para obtener la probabilidad de cumplir un plazo, el algoritmo siempre termina, incluso cuando el número de trabajos futuros sea infinito. Por otro lado, la probabilidad de incumplir un plazo es uno menos la probabilidad de cumplirlo, por lo que ésta también puede obtenerse de forma exacta tras un número finito de iteraciones.

En las secciones siguientes formalizaremos todas las ideas presentadas en este

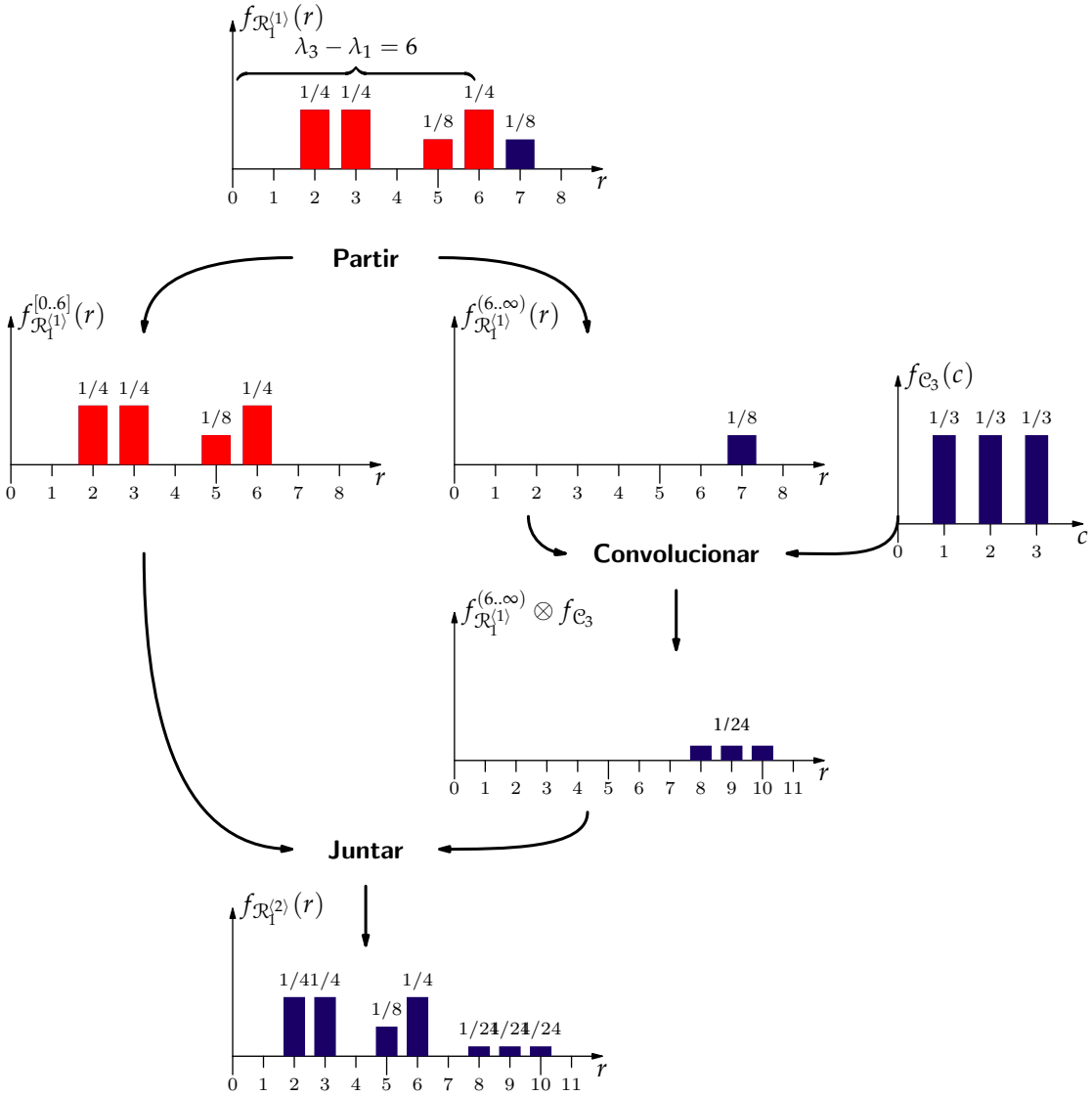


Figura 4.8.: Cálculo de la PF del tiempo de respuesta. Paso dos

ejemplo, dándoles la forma de un teorema y un algoritmo.

4.4.2. Expresión recursiva para la PF del tiempo de respuesta

Supongamos que tenemos una secuencia de trabajos $\{\Gamma_j\}$, con $j = 0, 1, \ldots$ ordenados por sus instantes de activación, esto es $\lambda_i \leq \lambda_j$ para $i < j$. Supongamos que la carga existente en el instante λ_0, $\mathcal{W}(\lambda_0)$, sigue una función de probabilidad conocida, y que el primer trabajo, Γ_0, es un trabajo basal (esto es, toda la carga existente en el instante en que este trabajo llega, causará retraso en el mismo). Queremos obtener la función de probabilidad del tiempo de respuesta de este trabajo Γ_0.

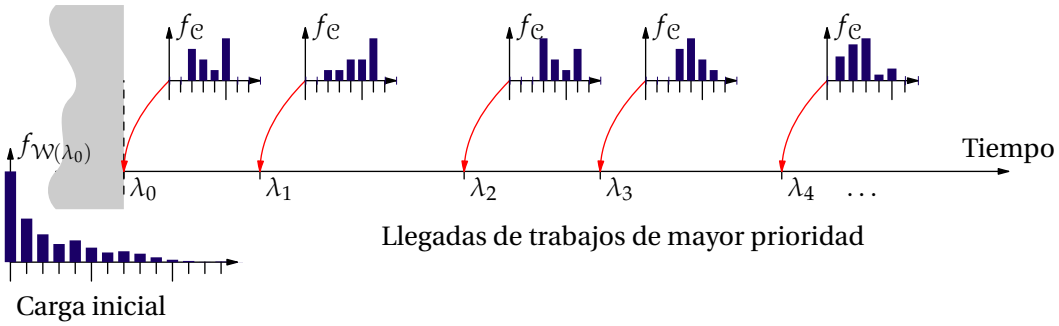


Figura 4.9.: Planteamiento del problema del cálculo del tiempo de respuesta

La figura 4.9 resume el planteamiento del problema. Obsérvese que, desde el momento que suponemos conocida la distribución de la carga en el instante λ_0, no necesitamos conocer las características de los trabajos que hubiera antes de λ_0 (si es que los hubo). La distribución de la variable aleatoria $\mathcal{W}(\lambda_0)$ de alguna forma "resume" todo el pasado del sistema (esto intenta representarse en la figura 4.9 mediante una zona gris). Sin pérdida de generalidad, podemos suponer que $\lambda_0 = 0$ (si no lo fuera, basta restar λ_0 a todos los instantes de activación, esto es, desplazar el origen de tiempos).

Por tanto, este planteamiento es genérico y aplicable a cualquier trabajo basal del sistema, aunque no sea el primero, ya que mediante el método de "convolucionar y encoger" podemos avanzar hasta un instante cualquiera, en particular hasta el instante de activación de cualquier trabajo, y calcular la carga en ese momento. A partir de aquí, podemos considerar a ese trabajo como el primero de una secuencia y aplicar el análisis que veremos ahora.

Para simplificar el desarrollo, y sin pérdida de generalidad, asumiremos que todos los trabajos futuros tienen prioridad mayor que el considerado, esto es $P_j > P_0$ para $j > 0$. Si no fuera así, eliminaríamos de la secuencia aquellos trabajos con prioridad menor que P_0, y usaríamos la nueva secuencia resultante en lugar de la original.

Basándonos en las ideas introducidas en el ejemplo preliminar, comenzaremos por definir formalmente la variable aleatoria $\mathcal{R}_0^{\langle k \rangle}$.

Definición 8. *Denotaremos por $\mathcal{R}_0^{\langle k \rangle}$ al tiempo de respuesta que tendría el trabajo Γ_0 si no sufriese interferencia (expulsión) de ningún trabajo Γ_j con $j > k$.*

Dicho de otro modo, sería su tiempo de respuesta si sólo pudieran expulsarle los primeros k trabajos que le siguen.

En particular, $\mathcal{R}_0^{\langle 0 \rangle}$ es el tiempo de respuesta que tendría el trabajo Γ_0 si ningún otro trabajo pudiera expulsarle. Puesto que este tiempo sería igual a su propio tiempo de ejecución, más la carga existente en el instante λ_0, y ambas son variables aleatorias, $\mathcal{R}_0^{\langle 0 \rangle}$ también será una variable aleatoria. Su función de probabilidad se obtendrá mediante la convolución de las funciones correspondientes. Es decir:

$$f_{\mathcal{R}_0^{\langle 0 \rangle}}(r) = \left(f_{\mathcal{W}(\lambda_0)} \otimes f_{\mathcal{C}_0}\right)(r) \tag{4.8}$$

Para el caso más general de $\mathcal{R}_0^{\langle k \rangle}$ no es posible escribir una fórmula sencilla que nos de su función de probabilidad, sino que deberá obtenerse en forma iterativa a partir de la función de probabilidad para $(k-1)$. El método será desarrollado formalmente en el siguiente teorema. A partir de él, y haciendo $k \to \infty$ tendremos la función de probabilidad del tiempo de respuesta $\mathcal{R}_0$, que tiene en cuenta el efecto de todos los trabajos futuros que le pueden interferir. Veremos también que no es necesario hacer $k \to \infty$ para obtener resultados útiles, ya que en cada nueva iteración obtendremos un "trozo" del resultado final, y podremos detener el algoritmo cuando el trozo obtenido sea lo bastante grande como para permitirnos responder a la pregunta ¿qué probabilidad hay de que el tiempo de respuesta sea menor que el plazo D_0?

Teorema 1. *Si la función de probabilidad de $\mathcal{R}_0^{\langle k-1 \rangle}$ es conocida, a partir de ella puede obtenerse la de $\mathcal{R}_0^{\langle k \rangle}$ mediante la siguiente fórmula:*

$$f_{\mathcal{R}_0^{\langle k \rangle}}(r) = f_{\mathcal{R}_0^{\langle k-1 \rangle}}^{[0..\lambda_k]}(r) + \left(f_{\mathcal{R}_0^{\langle k-1 \rangle}}^{(\lambda_k..\infty\rangle} \otimes f_{\mathcal{C}_k}\right)(r) \tag{4.9}$$

(véase definición 7, en página 70 para la explicación de la notación)

Demostración. Dividiremos la demostración en dos casos, según el valor de r sea menor o igual que λ_k o no.

Caso 1: $r \leq \lambda_k$. Este caso es bastante trivial, ya que para estos valores, la función $f_*^{(\lambda_k..\infty)}(r)$ vale cero por definición, de modo que en la ecuación (4.9) la convolución del segundo termino sale cero, y la ecuación se reduce a:

$$f_{\mathcal{R}_0^{\langle k \rangle}}(r) = f_{\mathcal{R}_0^{\langle k-1 \rangle}}^{[0..\lambda_k]}(r) \qquad r \leq \lambda_k$$

Por otro lado, para los valores $r \leq \lambda_k$, la función $f_*^{[0..\lambda_k]}(r)$ coincide por definición con $f_*(r)$, por lo que la anterior ecuación puede dejarse como:

$$f_{\mathcal{R}_0^{\langle k \rangle}}(r) = f_{\mathcal{R}_0^{\langle k-1 \rangle}}(r) \qquad r \leq \lambda_k$$

De acuerdo con la definición 8 en la página 76, la ecuación anterior puede leerse en la forma siguiente: "La probabilidad de que el tiempo de respuesta tome un valor r, estando r por debajo de λ_k, es la misma tanto si se toman en cuenta los trabajos que llegan después de λ_k como si no". Esto es cierto, puesto que si $r \leq \lambda_k$, el trabajo Γ_0 finalizará antes del instante λ_k, y por tanto los trabajos que lleguen después ya no podrán expulsarle y no podrán alterar su tiempo de respuesta.

Caso 2: $r > \lambda_k$. Cuando $r > \lambda_k$, el trabajo Γ_0 sufrirá interferencia por parte del trabajo Γ_k que llega en λ_k. Debemos incorporar el efecto de esta interferencia al tiempo de respuesta de Γ_0.

El tiempo de respuesta $\mathcal{R}_0^{\langle k-1 \rangle}$ es el que tendría el trabajo Γ_0 sin tener en cuenta el efecto que sobre él puedan causar los trabajos Γ_k, $\Gamma_{k+1}, \ldots$. El tiempo de respuesta $\mathcal{R}_0^{\langle k \rangle}$ ya toma en consideración el efecto de Γ_k (pero no el efecto de $\Gamma_{k+1}, \ldots$). La cuestión es cuál es la probabilidad de que $\mathcal{R}_0^{\langle k \rangle}$ tome un cierto valor $r > \lambda_k$. Se trata de la probabilidad de una intersección, ya que para que este evento ocurra deben cumplirse dos condiciones:

1. El trabajo Γ_0 debe estar aún activo en el instante λ_k. Esto implica que su tiempo de respuesta, incluso sin considerar el efecto de Γ_k, debe ser ya mayor de λ_k. Es decir, debe cumplirse la condición:

$$\mathcal{R}_0^{\langle k-1 \rangle} > \lambda_k$$

2. La suma del tiempo de respuesta antes de considerar la interferencia de Γ_k, más el propio tiempo de ejecución de Γ_k debe ser igual a r. Es decir, debe cumplirse también la condición:

$$\mathcal{R}_0^{\langle k-1 \rangle} + \mathcal{C}_k = r$$

La probabilidad de que ambas condiciones se cumplan simultáneamente es el sumatorio de las probabilidades de todos los eventos que están dentro de la intersección. Es decir, de entre todos los casos posibles, sumaremos aquellos que verifiquen ambas condiciones. Esto nos lleva a la ecuación:

$$\mathbb{P}\{\mathcal{R}_0^{\langle k \rangle} = r\} = \sum_{i > \lambda_k} \mathbb{P}\{\mathcal{R}_0^{\langle k-1 \rangle} = i\} \cdot \mathbb{P}\{\mathcal{C}_k = r - i\} \qquad \text{para } r > \lambda_k$$

Usando las funciones de probabilidad, la ecuación anterior puede reescribirse como:

$$\mathbb{P}\{\mathcal{R}_0^{\langle k\rangle}=r\} = \sum_{i>\lambda_k} f_{\mathcal{R}_0^{\langle k-1\rangle}}(i)\cdot f_{\mathcal{C}_k}(r-i) \qquad \text{para } r>\lambda_k$$

Ahora bien, en el rango $i > \lambda_k$, que es el rango del sumatorio, las funciones $f_*(i)$ y $f_*^{(\lambda_k..\infty)}(i)$ son idénticas, por lo que podemos escribir la segunda:

$$\mathbb{P}\{\mathcal{R}_0^{\langle k\rangle}=r\} = \sum_{i>\lambda_k} f_{\mathcal{R}_0^{\langle k-1\rangle}}^{(\lambda_k..\infty)}(i)\cdot f_{\mathcal{C}_k}(r-i) \qquad \text{para } r>\lambda_k$$

Finalmente, ya que la función $f_*^{(\lambda_k..\infty)}(i)$ es cero para todo $i \leq \lambda_k$, podemos eliminar el rango del sumatorio y dejar que éste se evalúe para todo i. El resultado seguirá siendo el mismo. Esto nos lleva a:

$$\mathbb{P}\{\mathcal{R}_0^{\langle k\rangle}=r\} = \sum_{i=-\infty}^{\infty} f_{\mathcal{R}_0^{\langle k-1\rangle}}^{(\lambda_k..\infty)}(i)\cdot f_{\mathcal{C}_k}(r-i) \qquad \text{para } r>\lambda_k$$

Ahora vemos que el segundo miembro de la ecuación representa la convolución de las funciones $f_{\mathcal{R}_0^{\langle k-1\rangle}}^{(\lambda_k..\infty)}(\cdot)$ y $f_{\mathcal{C}_k}(\cdot)$. Finalmente, podemos sumar a este segundo miembro la función $f_{\mathcal{R}_0^{\langle k-1\rangle}}^{[0..\lambda_k]}(r)$ sin que el resultado se altere, puesto que esta función es cero para $r > \lambda_k$. De modo que finalmente obtenemos:

$$\mathbb{P}\{\mathcal{R}_0^{\langle k\rangle}=r\} = f_{\mathcal{R}_0^{\langle k-1\rangle}}^{[0..\lambda_k]}(r) + \left(f_{\mathcal{R}_0^{\langle k-1\rangle}}^{(\lambda_k..\infty)} \otimes f_{\mathcal{C}_k}\right)(r) \qquad \text{para } r>\lambda_k$$

Esto completa la demostración de que la ecuación (4.9) es cierta también para el caso 2. □

Corolario 1. *La función de probabilidad del tiempo de respuesta del trabajo* Γ_0 *coincide en sus primeros* λ_k *puntos con la de la variable aleatoria* $\mathcal{R}_0^{\langle k-1\rangle}$*. Esto es:*

$$f_{\mathcal{R}_0}(r) = f_{\mathcal{R}_0^{\langle k-1\rangle}}(r) \qquad \textit{para } r \leq \lambda_k \tag{4.10}$$

Demostración. Como se ha demostrado en el *caso 1* del teorema, la función $f_{\mathcal{R}_0^{\langle k\rangle}}(\cdot)$ coincide en sus primeros λ_k puntos con la función $f_{\mathcal{R}_0^{\langle k-1\rangle}}(\cdot)$.

Análogamente, la función $f_{\mathcal{R}_0^{\langle k+1\rangle}}(\cdot)$ coincide en sus primeros λ_{k+1} puntos con la función $f_{\mathcal{R}_0^{\langle k\rangle}}(\cdot)$, y ya que $\lambda_{k+1} \geq \lambda_k$, coincidirá en sus λ_k puntos con $f_{\mathcal{R}_0^{\langle k-1\rangle}}(\cdot)$.

Este razonamiento puede repetirse, y se deduce que para cualquier j positivo, la función $f_{\mathcal{R}_0^{\langle k+j\rangle}}(\cdot)$ coincide en sus primeros λ_k puntos con la función $f_{\mathcal{R}_0^{\langle k-1\rangle}}(\cdot)$. Y puesto que $\mathcal{R}_0^{\langle\infty\rangle} = \mathcal{R}_0$ por definición, haciendo tender $j \to \infty$ tendremos la demostración del corolario. □

Corolario 2. *Si existe un* k *tal que* $f_{\mathcal{R}_0^{\langle k\rangle}}(r) = 0$ *para todo* $r > \lambda_{k+1}$*, entonces* $f_{\mathcal{R}_0}(r) = f_{\mathcal{R}_0^{\langle k\rangle}}(r)$ *para todo* r*.*

Demostración. El caso que se enuncia en este corolario significa que todos los posibles valores de $\mathcal{R}_0^{\langle k \rangle}$ son inferiores a λ_{k+1} (la probabilidad de que sea mayor es cero). Por tanto el trabajo finalizará antes de que pueda ser interrumpido por Γ_{k+1} y por tanto la función de probabilidad obtenida en el paso k coincidirá con la "verdadera" función de probabilidad del tiempo de respuesta $\mathcal{R}_0$.

Aunque la interpretación del corolario es clara, también puede demostrarse matemáticamente de forma muy sencilla. En el caso enunciado, la operación de "partir en dos" la función $f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$, produciría como resultado una mitad superior nula (cero en todos sus puntos). Por tanto, en la fórmula (4.9) la convolución del segundo término saldrá cero, y por tanto se obtendrá que $f_{\mathcal{R}_0^{\langle k+1 \rangle}}(\cdot)$ sale igual que $f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$. Este razonamiento se repetirá para todas las iteraciones siguientes, y en todas ellas se obtendrá una y otra vez el mismo resultado. Es decir, para cualquier j positivo se tendrá que $f_{\mathcal{R}_0^{\langle k+j \rangle}}(\cdot) = f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$. Haciendo tender $j \to \infty$ se demuestra este corolario. □

Corolario 3. *La probabilidad de que el trabajo Γ_0 pierda su plazo D_0 puede calcularse encontrando el entero k tal que $\lambda_k < D_0 \leq \lambda_{k+1}$, y aplicando entonces la fórmula*

$$\mathbb{P}\{\mathcal{R}_0 > D_0\} = 1 - \sum_{i=0}^{D_0} f_{\mathcal{R}_0^{\langle k \rangle}}(i) \tag{4.11}$$

Demostración. En realidad, la probabilidad de perder el plazo es igual a 1 menos la probabilidad de cumplirlo. Y la probabilidad de cumplirlo es la suma de probabilidades para los casos en los que el tiempo de respuesta sea menor o igual que el plazo. Por tanto:

$$\mathbb{P}\{\mathcal{R}_0 > D_0\} = 1 - \sum_{i=0}^{D_0} \mathbb{P}\{\mathcal{R}_0 = i\} = 1 - \sum_{i=0}^{D_0} f_{\mathcal{R}_0}(i)$$

De acuerdo con el corolario 1, la función $f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$ coincide con $f_{\mathcal{R}_0}(\cdot)$ en sus primeros λ_{k+1} puntos. Por tanto, si $D_0 \leq \lambda_{k+1}$, podemos sustituir $f_{\mathcal{R}_0}(\cdot)$ por $f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$, en el último sumatorio, lo que demuestra el corolario. □

4.4.3. Algoritmo

El teorema anterior y sus corolarios tienen importantes implicaciones prácticas. En primer lugar, el teorema 1 nos proporciona un algoritmo iterativo para obtener la función de probabilidad de $\mathcal{R}_0^{\langle k \rangle}$ para un k cualquiera. En efecto, partiendo de la función de probabilidad de $\mathcal{R}_0^{\langle 0 \rangle}$, que se calcula siguiendo la ecuación (4.8), podemos obtener la de $\mathcal{R}_0^{\langle 1 \rangle}$, y a partir de ésta la de $\mathcal{R}_0^{\langle 2 \rangle}$, y así sucesivamente hasta llegar a cualquier $\mathcal{R}_0^{\langle k \rangle}$ deseado.

En segundo lugar, de acuerdo con el corolario 1, cada función $f_{\mathcal{R}_0^{\langle k \rangle}}(\cdot)$ obtenida coincide con la buscada $f_{\mathcal{R}_0}(\cdot)$ en sus primeros λ_{k+1} puntos. Por tanto, si estamos interesados en conocer la probabilidad de que $\mathcal{R}_0$ tome un valor r dado, basta iterar hasta alcanzar un k tal que $r \leq \lambda_{k+1}$. El valor de $f_{\mathcal{R}_0^{\langle k \rangle}}(r)$ nos dará la respuesta buscada. Es decir, para obtener la probabilidad de un tiempo de respuesta dado, el número de iteraciones a realizar será siempre finito. Solamente si queremos la completa función de probabilidad del tiempo de respuesta el número de iteraciones puede llegar a ser infinito. Pero de acuerdo con el corolario 2 puede alcanzarse una condición de parada en la cual no sea necesario realizar más iteraciones. Más adelante estudiaremos bajo qué condiciones podemos garantizar que esta condición de parada va a aparecer.

Por último, en el caso particular de que lo que se busque sea simplemente la probabilidad de pérdida del plazo, el corolario 3 nos dice que podremos obtenerla siempre tras un número finito de iteraciones (k).

Podemos por tanto enunciar el siguiente algoritmo para obtener la función f de probabilidad del tiempo de respuesta de un trabajo cualquiera Γ_j:

1: Inicializar: $f \leftarrow f_{\mathcal{W}(\lambda_j)} \otimes \mathcal{C}_j$
2: $k \leftarrow 1$
3: **mientras** $(\lambda_k - \lambda_j) < D_j$ **hacer**
4: Partir:
$f_1 \leftarrow$ primeros $(\lambda_k - \lambda_j)$ puntos de f
$f_2 \leftarrow$ restantes puntos de f
5: **si** no hay puntos en f_2 **entonces**
6: Salir del bucle
7: **fin si**
8: Convolucionar: $f_{\text{aux}} \leftarrow f_2 \otimes f_{\mathcal{C}_k}$
9: Juntar: $f \leftarrow f_1 + f_{\text{aux}}$
10: $k \leftarrow k + 1$
11: **fin mientras**

Del algoritmo se puede salir por dos razones diferentes, bien por la instrucción 6, o bien por que la condición de la línea 3 deja de ser cierta. En el primer caso, la f final será la función de probabilidad completa y exacta de $\mathcal{R}_j$. En el segundo caso, se obtiene una función incompleta, que es válida sólo en el rango $[0, \lambda_k - \lambda_j)$, pero que es útil para obtener la probabilidad de cumplir el plazo (y por tanto la probabilidad de perderlo).

En la sección siguiente ejemplificaremos el uso de este algoritmo, y del método "convolucionar y encoger" presentado en la sección 4.3.1, aplicándolo a un caso clásico de la literatura.

4.4.4. Ejemplo de aplicación

Presentamos a continuación un ejemplo con dos tareas periódicas. Este ejemplo está basado en el presentado por Lehoczky [1990] para ilustrar el concepto de periodo ocupado (*busy-period*). El ejemplo original tenía dos tareas, de periodos 70 y 100, y tiempos de computación fijos de 26 y 62 unidades, respectivamente. La política de asignación de prioridades *rate-monotonic* asigna una prioridad mayor a la primera tarea por tener menor periodo. En este sistema, la segunda tarea presentaba un tiempo de respuesta para el peor caso de 118 unidades, como se calcula en [Lehoczky, 1990]. Debido a que los plazos de las tareas eran mayores que sus periodos, una tarea podía causar interferencia sobre sí misma, y esto implica que el peor tiempo de respuesta ya no ocurre necesariamente en la primera activación de la tarea. De hecho, en este ejemplo ocurría en la quinta activación.

Usaremos el mismo conjunto de tareas, con los mismos periodos, pero supondremos ahora que los tiempos de computación, en lugar de ser deterministas, son aleatorios. El tiempo de computación de la primera tarea puede tomar dos valores, 25 ó 26, con igual probabilidad, y el de la segunda tarea puede ser 61 ó 62, también con igual probabilidad. Esta información se resume en el cuadro 4.3.

Tarea	T_i	Prioridad	$f_{\mathcal{C}_i}(\cdot)$
τ_1	70	alta	U[25,26]
τ_2	100	baja	U[61,62]

Cuadro 4.3.: Un sencillo sistema con dos tareas periódicas

El análisis de la tarea τ_1 es trivial, puesto que, al ser la de mayor prioridad, no puede sufrir interferencias de otras tareas. La PF de su tiempo de respuesta coincide con la PF de su tiempo de ejecución.

Analicemos el tiempo de respuesta de la tarea τ_2. Esta tarea se activa varias veces y en general en cada activación presentará un perfil diferente de tiempo de respuesta (puesto que la carga existente cuando llega es diferente, así como las interferencias que sufrirá). En principio, sería necesario obtener la función de probabilidad para cada una de sus activaciones (trabajos), y después promediar todas ellas para obtener la de la tarea. En este ejemplo calcularemos con detalle el caso de su quinta activación, es decir, calcularemos el tiempo de respuesta del trabajo $\Gamma_{2,5}$. Para otras activaciones se procedería de forma análoga.

Carga existente cuando se produce la quinta activación

El primer paso será obtener la distribución de la carga en el instante de esta quinta activación. La figura 4.10 muestra la secuencia de activaciones. Para obte-

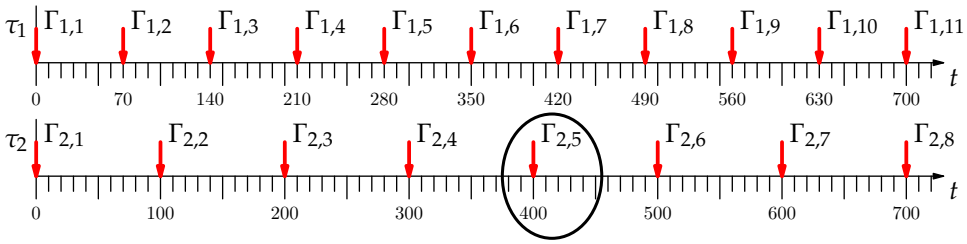


Figura 4.10.: Secuencia de activaciones de trabajos en el ejemplo

ner la carga existente en el instante $\lambda_{2,5}$, justo antes de la llegada del trabajo $\Gamma_{2,5}$, será necesario partir de la carga inicial del sistema y aplicar repetidamente el algoritmo "convolucionar y encoger", hasta llegar a dicho instante. Observando la figura 4.10 vemos que será necesario aplicar 10 veces el método, puesto que hay 10 trabajos cuya activación se produce antes del instante $\lambda_{2,5}$.

La figura 4.11 muestra la función de probabilidad de la carga $\mathcal{W}(t)$ en diferentes activaciones de trabajos. La carga inicial del sistema se asume igual a cero con probabilidad uno. Vemos que poco después del instante cero, la carga ya ha aumentado, debido a la llegada de los trabajos $\Gamma_{1,1}$ y $\Gamma_{1,2}$. Ahora la carga puede tener tres diferentes valores (86, 87 u 88), con diferentes probabilidades. Estos son el resultado de la convolución de los tiempos de ejecución de los primeros trabajos. Avanzamos hasta el instante 70, en que llega el siguiente trabajo. Para ello, tenemos que "encoger" 70 unidades la función. En este caso es un mero desplazamiento sin acumulación en el origen, puesto que todos los valores de la función original estaban por encima de 70. Después, para pasar al instante 70^+, basta hacer la convolución con el tiempo de ejecución del trabajo $\Gamma_{1,2}$ que llega en el instante 70.

Así vamos convolucionando y encogiendo, hasta que llegamos al instante 400 que nos ocupa. Vemos que el trabajo anterior había llegado en 350, por lo que el último paso "encoger" es de 50 unidades. En este caso sí se produce una acumulación en el origen, como muestra la última gráfica de la figura 4.11. En esta gráfica puede leerse que existe una probabilidad de 0.744 de que la carga sea cero. Ya que en la figura apenas se aprecian los valores pequeños, en el cuadro 4.4 se muestran los valores numéricos. Vemos que la carga sólo puede tomar valores entre 0 y 4. Cualquier valor por encima de 4 tiene probabilidad cero. Esta es la carga que encuentra el trabajo $\Gamma_{2,5}$, y por tanto el retraso que podrá experimentar cuando inicie su ejecución.

Cálculo del tiempo de respuesta

Para obtener la distribución del tiempo de respuesta del trabajo $\Gamma_{2,5}$, aplicaremos ahora el algoritmo de la página 80.

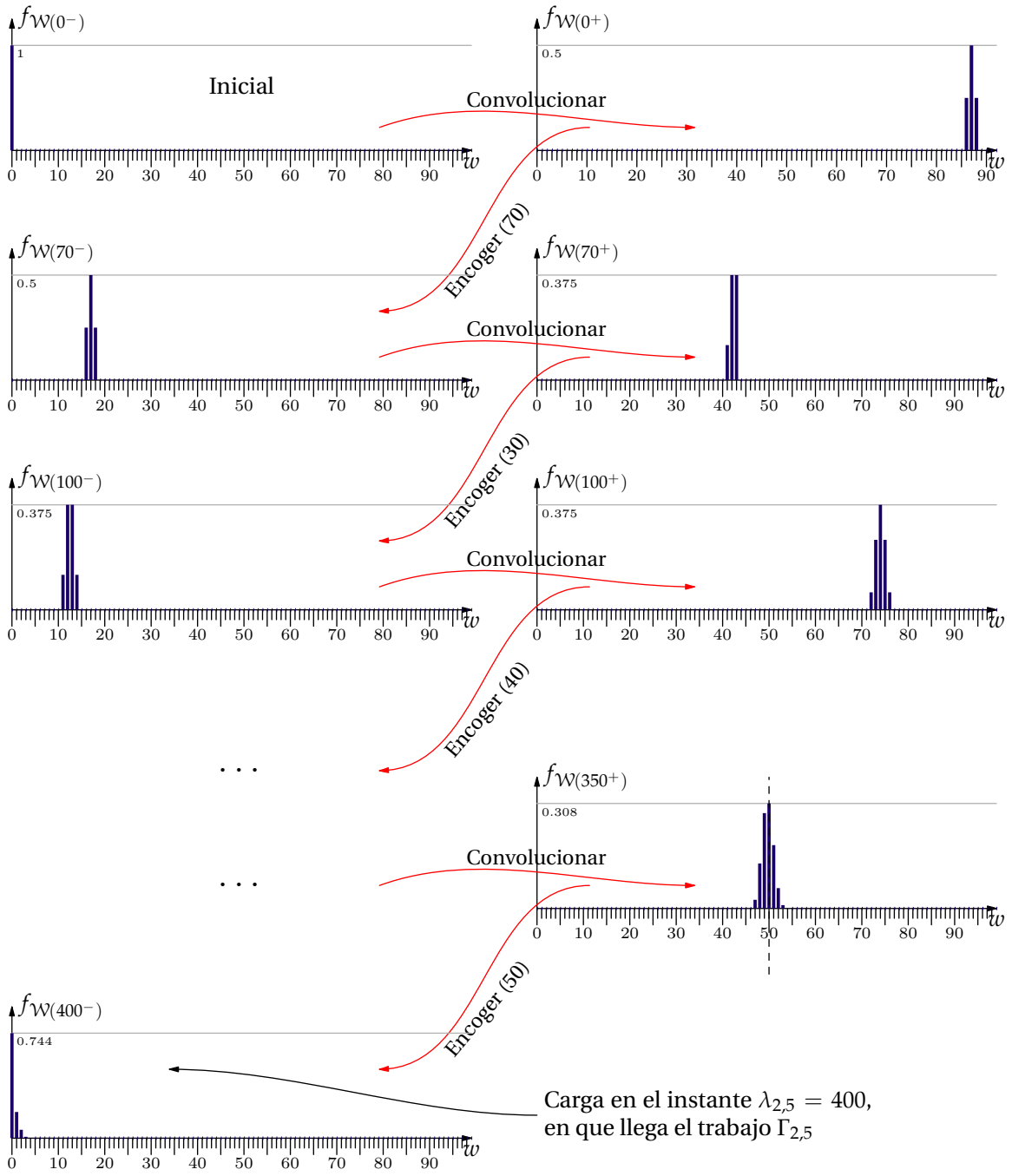


Figura 4.11.: Distribución estadística de la carga del sistema de ejemplo en diferentes instantes

w	$\mathbb{P}\{\mathcal{W}(400) = w\}$
0	0.74414
1	0.18555
2	0.05957
3	0.00977
4	0.00098

Cuadro 4.4.: Posibles valores de la carga y sus probabilidades para el instante $\lambda_{2,5} = 400$

- Inicialización: En primer lugar tenemos que redefinir el sistema de modo que se ajuste a las hipótesis del teorema. Todos los trabajos que lleguen después de $\Gamma_{2,5}$ han de tener mayor prioridad, por lo que tendremos que eliminar del sistema todos los que corresponden a la tarea τ_2, dejando sólo los de τ_1. Una vez hecho esto, numeramos estos trabajos por orden de activación, y fijamos el nuevo origen de tiempos en el instante $\lambda_{2,5}$ (restando 400 a todos los instantes). El nuevo sistema resultante se muestra en la figura 4.12. Observar cómo todo el pasado del sistema se resume en la distribución de la carga. No nos importa cómo se ha generado esa carga (es decir, qué trabajos han llegado antes y en qué orden), sino tan sólo su función de probabilidad.

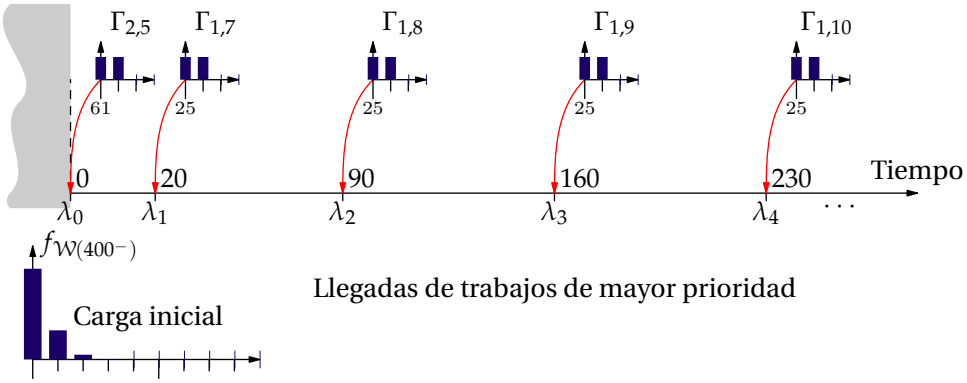


Figura 4.12.: Sistema tras la transformación necesaria para aplicarle el algoritmo

- Calculamos la distribución de $\mathcal{R}_0^{\langle 0 \rangle}$, mediante la convolución de la carga con el tiempo de ejecución de éste trabajo. El resultado se muestra en la figura 4.13.

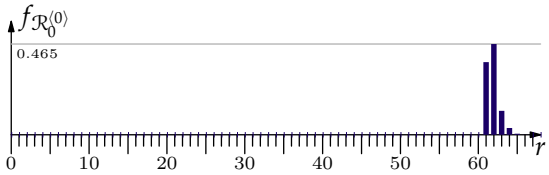


Figura 4.13.: Función de probabilidad de $\mathcal{R}_0^{\langle 0 \rangle}$

- Para calcular la distribución de $\mathcal{R}_0^{\langle 1 \rangle}$ partimos en dos la de $\mathcal{R}_0^{\langle 0 \rangle}$, usando como punto de división el valor 20 (que es el valor de λ_1). Convolucionamos la parte alta con el tiempo de ejecución de Γ_1, que es el de la tarea τ_1, y juntamos el resultado con la parte baja. Esta operación se muestra en la figura 4.14(a)
- Se procede análogamente, partiendo en dos la función de probabilidad de $\mathcal{R}_0^{\langle 1 \rangle}$ (ahora el punto de corte es $\lambda_2 = 90$), convolucionamos la parte alta con la función de probabilidad del siguiente trabajo, y juntamos el resultado con la parte baja. Esta operación se muestra en la figura 4.14(b).
- Si ahora intentamos aplicar de nuevo el algoritmo, habría que partir la función de probabilidad de $\mathcal{R}_0^{\langle 2 \rangle}$ usando como punto de corte $\lambda_3 = 160$. Sin embargo vemos que a la derecha de ese punto, todos los valores son cero, lo que indica que es imposible que los siguientes trabajos puedan causar interferencia. Por tanto el algoritmo ha finalizado y el resultado es:

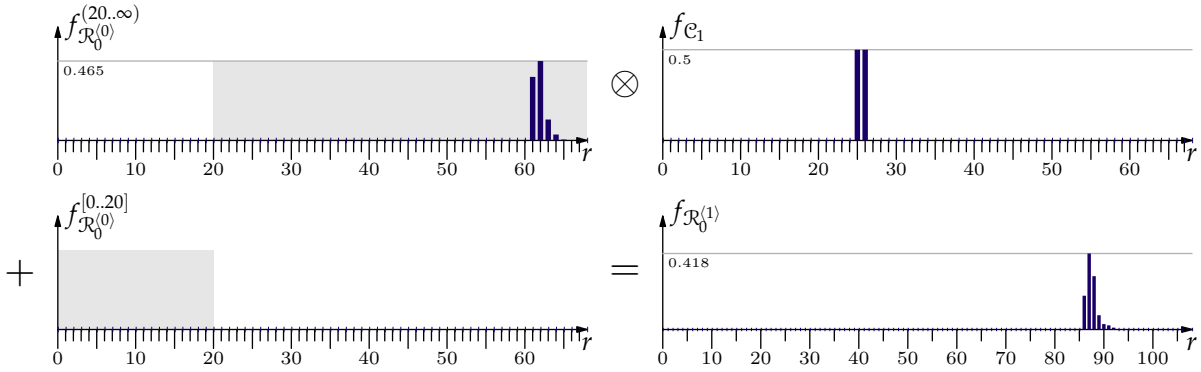
$$f_{\mathcal{R}_0}(r) = f_{\mathcal{R}_0^{\langle 2 \rangle}}(r) \qquad \text{para todo } r$$

Puesto que en la figura apenas se aprecian los puntos de la derecha, reproducimos sus valores numéricos en el cuadro 4.5.

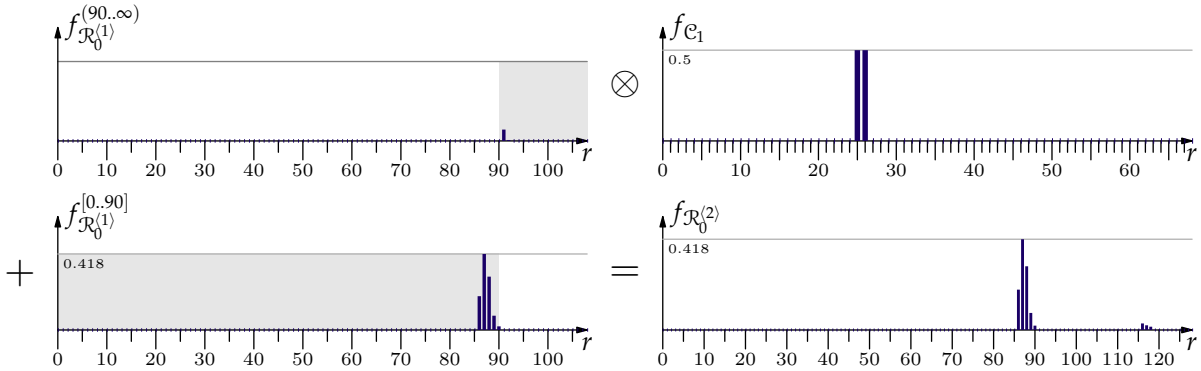
r	$\mathbb{P}\{\mathcal{R}=r\}$	r	$\mathbb{P}\{\mathcal{R}=r\}$	r	$\mathbb{P}\{\mathcal{R}=r\}$
86	0.186035	97	0	108	0
87	0.418457	98	0	109	0
88	0.293701	99	0	110	0
89	0.078613	100	0	111	0
90	0.020020	101	0	112	0
91	0	102	0	113	0
92	0	103	0	114	0
93	0	104	0	115	0
94	0	105	0	116	0.001465
95	0	106	0	117	0.001587
96	0	107	0	118	0.000122

Cuadro 4.5.: Posibles valores del tiempo de respuesta y sus probabilidades, para la quinta activación de la tarea τ_2

En este cuadro observamos unos cuantos hechos interesantes. En primer lugar, el valor máximo que el tiempo de respuesta puede alcanzar es de 118 unidades. Este resultado concuerda con el del análisis clásico. Ahora bien, nuestro análisis aporta información adicional, ya que nos dice que la probabilidad de que este peor tiempo de respuesta aparezca es tan sólo del 0.0122 %. De hecho, en el 99.6826 % de los casos, el tiempo de respuesta estará por debajo de las 91 unidades. También podemos ver que el tiempo de respuesta más frecuente es el de 87



(a) Cálculo de la distribución de $\mathcal{R}_0^{\langle 1\rangle}$



(b) Cálculo de la distribución de $\mathcal{R}_0^{\langle 2\rangle}$

Figura 4.14.: Resultados de las diferentes iteraciones del algoritmo

unidades y que hay una serie de tiempos de respuesta "prohibidos" (con probabilidad cero) entre los valores 91 y 115. El valor esperado del tiempo de respuesta puede calcularse con la fórmula siguiente:

$$\bar{R_0} = \sum_{i=0}^{\infty} i \cdot \mathbb{P}\{\mathcal{R}_0 = i\}$$

En nuestro caso, el sumatorio se evalúa sólo para i entre 86 y 118 (puesto que fuera de ese rango las probabilidades son cero), y se obtiene como resultado $\bar{R}_i = 87{,}4188$.

Observar que todos estos resultados estadísticos se aplican sólo a la quinta activación de la tarea τ_2. Esta tarea tiene otras activaciones y en cada una de ellas la distribución de tiempos de respuesta será en general diferente. Por tanto, es necesario repetir un análisis como el anterior para cada una de sus activaciones.

En este ejemplo concreto, la utilización máxima del sistema es inferior a 1 (en concreto $U^{\max} = 0{,}991429$), por lo que al final del hiperperiodo, justo antes del instante $t = 700$, la carga en el sistema será cero. Esto puede verificarse aplicando el algoritmo "convolucionar y encoger" hasta llegar a dicho instante. Se observa que el resultado es una función de probabilidad igual a la inicial, es decir, cero en todos sus puntos excepto en cero, que vale 1. Esto implica que el siguiente hiperperiodo comenzará exactamente con las mismas condiciones iniciales que el que acabamos de analizar. Por tanto la quinta activación dentro de ese hiperperiodo tendrá de nuevo la misma distribución que ya hemos calculado y mostrado en el cuadro 4.5.

En este ejemplo, en todos los hiperperiodos se repetirán las mismas distribuciones de los tiempos de respuesta, por lo que para obtener el promedio de todos los trabajos, basta promediar los de un hiperperiodo. Volviendo a la figura 4.10, vemos que la tarea τ_2 se activa 7 veces dentro del hiperperiodo, por lo que basta analizar el tiempo de respuesta de sus 7 primeras activaciones y promediarlos para obtener el tiempo de respuesta de la tarea. Es decir:

$$f_{\mathcal{R}_2}(r) = \frac{1}{7}\sum_{i=1}^{7} f_{\mathcal{R}_{2,i}}(r)$$

El cuadro 4.6 muestra las distribuciones de los tiempos de respuesta de cada una de estas 7 activaciones (los guiones representan probabilidad nula), y en la última columna el promedio de todas ellas. Esta última columna, por tanto, nos da la función de probabilidad del tiempo de respuesta de la tarea τ_2 y a partir de ella puede calcularse la probabilidad de que la tarea pierda su plazo, para cualquier plazo dado. Observar que la información sobre los plazos de las tareas no ha sido utilizada durante el análisis y que por tanto el análisis es aplicable tanto a tareas con plazos iguales a los periodos, como a tareas con plazos superiores a sus periodos.

Para solucionar el ejemplo anterior ha bastado con analizar los trabajos que se ejecutan en el primer hiperperiodo, puesto que en los siguientes hiperperiodos el análisis produciría el mismo resultado. Esto se debe a que la distribución de la carga en el instante 700 es idéntica a la del instante cero. Sin embargo, esta condición no se cumplirá necesariamente para todos los sistemas. En la sección siguiente analizaremos bajo qué condiciones se cumple, y cómo tratar el problema más general en el que la distribución de la carga vaya cambiando de un hiperperiodo al siguiente.

	Activación							
r	1	2	3	4	5	6	7	Promedio
86	—	—	—	—	0.186035	—	—	0.031006
87	—	—	—	—	0.418457	—	0.031151	0.074935
88	—	—	—	—	0.293701	—	0.155846	0.074925
89	—	—	—	—	0.078613	—	0.311974	0.065098
90	—	—	—	—	0.020019	—	0.312462	0.055414
91	—	—	—	—	—	—	0.156746	0.026124
92	—	—	—	—	—	—	0.031685	0.005281
93	—	—	—	—	—	—	0.000130	0.000022
94	—	—	—	—	—	—	0.000008	0.000001
95	—	—	—	—	—	—	—	—
96	—	—	—	—	—	—	—	—
97	—	0.031250	—	0.025391	—	—	—	0.009440
98	—	0.156250	—	0.131836	—	—	—	0.048014
99	—	0.312500	—	0.279297	—	—	—	0.098633
100	—	0.312500	—	0.307617	—	—	—	0.103353
101	—	0.156250	—	0.185547	—	0.124603	—	0.077733
102	—	0.031250	—	0.059570	—	0.374176	—	0.077499
103	—	—	—	0.009766	—	0.374939	—	0.064117
104	—	—	—	0.000977	—	0.125793	—	0.021128
105	—	—	—	—	—	0.000458	—	0.000076
106	—	—	—	—	—	0.000031	—	0.000005
107	—	—	—	—	—	—	—	—
108	—	—	—	—	—	—	—	—
109	—	—	—	—	—	—	—	—
110	—	—	—	—	—	—	—	—
111	0.125000	—	0.101562	—	—	—	—	0.037760
112	0.375000	—	0.324219	—	—	—	—	0.116537
113	0.375000	—	0.367188	—	—	—	—	0.123698
114	0.125000	—	0.171875	—	—	—	—	0.049479
115	—	—	0.031250	—	—	—	—	0.005208
116	—	—	0.003906	—	0.001465	—	—	0.000895
117	—	—	—	—	0.001587	—	—	0.000264
118	—	—	—	—	0.000122	—	—	0.000020

Cuadro 4.6.: Posibles valores del tiempo de respuesta y sus probabilidades para todas las activaciones de la tarea τ_2

4.5. Cálculo de la carga en régimen estacionario. Análisis Markoviano

4.5.1. Conceptos previos

Sea T la longitud del hiperperiodo, definida como se vio en la ecuación (4.1). Podemos dividir el tiempo en "zonas" de longitud T, comenzando en cero, y así marcamos como instantes de inicio de un hiperperiodo los instantes kT, siendo $k = 0, 1, 2, \ldots$ (ver figura 4.15). Ahora bien, como se observa en la figura, debido a que cada tarea tiene diferentes *offsets*, en el primer o los primeros hiperperiodos el patrón de activaciones de los trabajos puede ser diferente al de los hiperperiodos siguientes.

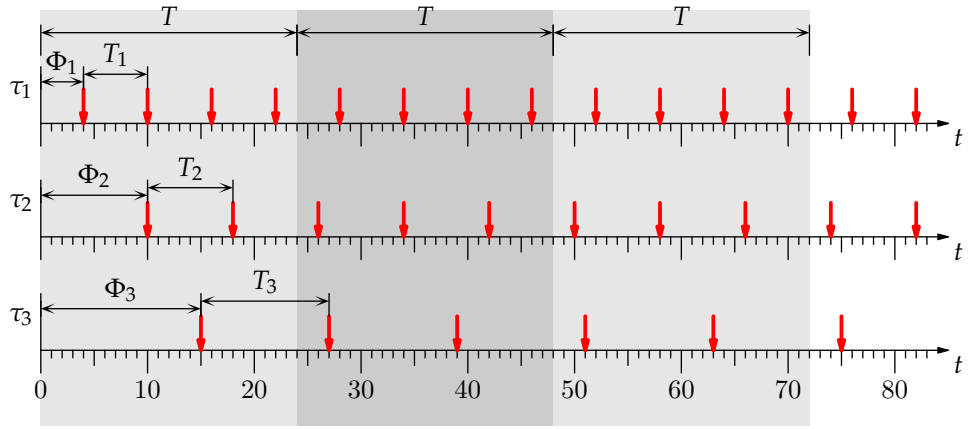


Figura 4.15.: Ejemplo de irregularidad en los primeros hiperperiodos

Debido a esta situación resulta conveniente la definición siguiente:

Definición 9. *Llamaremos **hiperperiodo completo** al intervalo de tiempo del tipo* $[kT,(k+1)T)$ *tal que en su interior cada tarea* τ_i *se activa* T/T_i *veces, siendo* T *la longitud del hiperperiodo y* T_i *el periodo de la tarea.*

*Llamaremos **primer hiperperiodo completo** al hiperperiodo completo con* k *mínima. Denotaremos por* k_0 *a esta* k *mínima, de modo que el primer hiperperiodo completo es el intervalo de tiempo* $[k_0T,(k_0+1)T]$.

En el ejemplo de la figura 4.15, el primer hiperperiodo completo es el que se inicia en $t = 24$, y por tanto en este caso $k_0 = 1$.

Está claro que antes del primer hiperperiodo completo tenemos una especie de "régimen transitorio", en el que el patrón de activaciones de trabajos no es periódico. A partir del primer hiperperiodo completo entramos en un "régimen permanente de activaciones" en el cual todos los siguientes hiperperiodos serán completos y el patrón de activación de los trabajos es exactamente el mismo en todos ellos.

Sin embargo, incluso el hecho de tener el mismo patrón de activación no garantiza que tengamos el mismo comportamiento estocástico (esto es, las mismas distribuciones en el tiempo de respuesta), ya que éste depende en general de todo el pasado del sistema. Este pasado se "resume" en la distribución de la carga, y ésta es una variable aleatoria como hemos visto cuya distribución varía en el tiempo. Sólo si esta distribución presenta un patrón repetitivo, el comportamiento estocástico del sistema también será repetitivo.

De modo que estamos interesados en observar la variable aleatoria $\mathcal{W}(t)$ en los instantes $t = kT, k = 0,1,2,\ldots$. Para simplificar la notación haremos uso de otra definición.

Definición 10. *Denotaremos por $\mathcal{W}_k$ a la carga existente en el instante kT, es decir, a la variable aleatoria $\mathcal{W}(kT)$, y nos referiremos a ella como "carga inicial del hiperperiodo k".*

4.5.2. Caso trivial: $U^{\max} \leq 1$

En el siguiente teorema demostramos que cuando la utilización máxima es menor que 1, la distribución de la carga se repetirá al principio de cada hiperperiodo y por tanto el análisis se podrá realizar en un hiperperiodo (como se hizo en el ejemplo de la sección 4.4.4).

Teorema 2. *Si en el instante inicial ($t = 0$) la carga es cero y además el sistema cumple $U^{max} \leq 1$, entonces la distribución de la carga al final del primer hiperperiodo completo ($t = (k_0 + 1)T$), se repetirá al final de todos los hiperperiodos siguientes.*

Demostración. Imaginemos el escenario en el que todos los trabajos que llegan requieren su máximo tiempo de ejecución. Llamemos $W^{\max}$ a la carga observada al final del primer hiperperiodo completo bajo estas circunstancias, y llamemos s al número de unidades de tiempo durante las cuales el sistema está ocioso en este primer hiperperiodo completo (nos referiremos a s como el "mínimo tiempo libre"). Es evidente que este tiempo libre, más el trabajo generado en este hiperperiodo, ha de ser igual a la longitud del hiperperiodo más la carga final, como se explica en la figura 4.16. Por otro lado, el trabajo generado en el hiperperiodo para este escenario de peor caso es igual a $TU^{\max}$, por lo que:

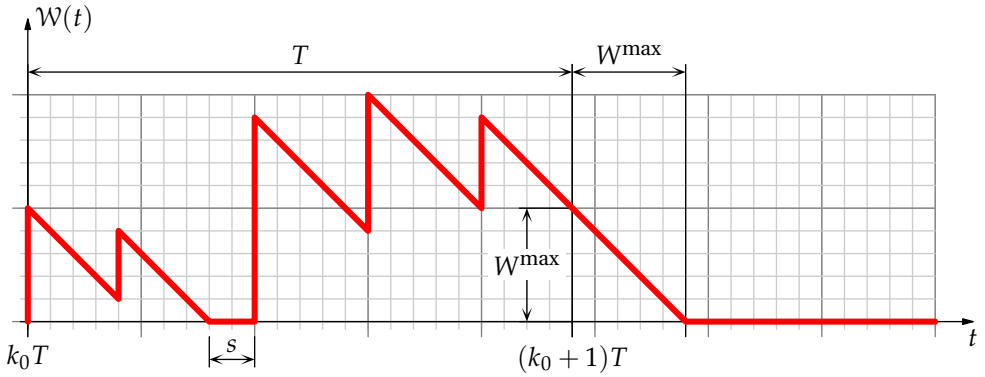
$$s + TU^{\max} = T + W^{\max}$$

Por otro lado, ya que por hipótesis $U^{\max} \leq 1$, esto implica que $TU^{\max} \leq T$ y llevando esto a la ecuación anterior se deduce que $W^{\max} \leq s$.

Si $W^{\max}$ fuese cero, la prueba sería trivial, ya que en ese caso el siguiente hiperperiodo comenzaría con carga cero, y ya que la suma de trabajos en él, incluso en el peor caso, es inferior a T, todos los hiperperiodos siguientes comenzarían también con carga cero.

Si $W^{\max}$ es positivo, esto sólo puede ocurrir si todo el tiempo libre s ocurrió antes del último periodo ocupado[4], como se comprende mirando la figura 4.15. Por tanto, el siguiente hiperperiodo "absorberá" esta carga $W^{\max}$ antes de llegar a su último periodo ocupado (debido a que $W^{\max} \leq s$ como vimos). Consiguientemente, la distribución de la carga al final del hiperperiodo no será afectada por la carga inicial del mismo, y será la misma que en el hiperperiodo anterior, y por tanto se repetirá idénticamente al final de todos los hiperperiodos siguientes. □

[4] Se denomina así al periodo de tiempo contiguo durante el cual la carga es positiva



En todo instante t se cumple que la suma de trabajos hasta ese instante (trazos verticales) más la suma de tiempo ocioso hasta ese instante (trazos horizontales), es igual a t más la carga para ese instante. Para simplificar, la figura no muestra los trabajos que llegan en el siguiente hiperperiodo.

Figura 4.16.: Prueba gráfica para el teorema 2

El teorema anterior nos proporciona la justificación para analizar un sólo hiperperiodo. En efecto, ya que la carga inicial de un hiperperiodo es la carga final del precedente, todos los hiperperiodos, a partir del segundo completo, presentan la misma distribución estadística de la carga inicial. Por tanto, el análisis del tiempo de respuesta producirá el mismo resultado en todos ellos. Los primeros hiperperiodos no-completos no son representativos con respecto al régimen permanente, por lo que no es necesario analizarlos.

Por tanto el algoritmo general para resolver este tipo de sistemas sería:

1. Calcular el instante de comienzo del primer hiperperiodo completo, esto es, el valor de k_0. Este valor depende exclusivamente de los *offsets* de las tareas. Observar que si el *offset* es cero para todas, el primer hiperperiodo completo comienza precisamente en $t = 0$, y por tanto $k_0 = 0$.

2. Calcular la distribución de la carga en el instante $(k_0 + 1)T$, utilizando el método "convolucionar y encoger". Observar que si todas las tareas tienen *offset* cero, el resultado será cero, por lo que este paso puede omitirse.

3. Realizar el análisis sobre un único hiperperiodo, utilizando como carga inicial la obtenida en el paso previo.

4.5.3. Caso general

Para el caso general, supondremos que $U^{\max} > 1$, puesto que si $U^{\max} \leq 1$ la solución se obtiene de forma más sencilla haciendo uso del método explicado en la sección 4.5.2.

Cuando $U^{\max} > 1$, existe una probabilidad no nula de que la carga generada en un hiperperiodo sea mayor que la propia longitud del hiperperiodo. Esto significa que existe una probabilidad no nula de que en el nuevo hiperperiodo la carga inicial sea mayor que en el hiperperiodo anterior. Esto implica por tanto que la distribución de la carga es diferente de un hiperperiodo a otro.

Demostraremos en esta sección que la carga observada a intervalos de longitud T, es decir, la secuencia de variables aleatorias $\{\mathcal{W}_i\}$, constituye una *cadena de Markov*. Encontraremos que la condición para que dicha cadena sea estable es que la utilización media, $\bar{U}$ sea menor que uno. Si esta condición se cumple, la distribución de la carga de cada $\mathcal{W}_i$ converge a medida que aumenta i hacia una distribución estacionaria. Encontraremos la forma general de esta distribución y daremos métodos para calcularla numéricamente.

La carga inicial es una cadena de Markov

Una cadena de Markov se define como una secuencia de variables aleatorias, tal que la PF de cada una de ellas depende únicamente de la PF de la predecesora, y no de las anteriores. Esta propiedad suele denominarse también la "falta de memoria" de la cadena de Markov.

La función de probabilidad de una variable aleatoria discreta $\mathcal{X}$ puede almacenarse en forma de vector, $\boldsymbol{x}$, de infinitas componentes, tal que la componente i-ésima de dicho vector sea la probabilidad de que $\mathcal{X}$ tome el valor i. Esta notación nos permite escribir la propiedad Markoviana en forma de ecuación matricial. Si la secuencia $\{\mathcal{X}_k\}$ de variables aleatorias es una cadena de Markov, entonces se cumple que

$$\boldsymbol{x}_{k+1} = \mathbf{P}\boldsymbol{x}_k$$

Siendo $\boldsymbol{x}_k$ la PF de la variable $\mathcal{X}_k$, almacenada en un vector como se ha explicado antes y $\mathbf{P}$ la llamada "matriz de transición", o también "matriz de Markov". Como vemos, conocida la PF de una de las variables de la secuencia, podemos obtener la de la siguiente variable multiplicando por la matriz $\mathbf{P}$. Y a partir de ella, la siguiente, etcétera. La clave para que la secuencia de $\mathcal{X}_k$ sea una cadena de Markov, es que la matriz $\mathbf{P}$ sea siempre la misma y no dependa de k. Veremos a continuación que este es el caso para la variable aleatoria $\mathcal{W}_k$, es decir, la carga inicial del hiperperiodo k-ésimo.

Definición 11. *Llamaremos* ***proceso de la carga*** *del sistema S a la secuencia de variables aleatorias $\{\mathcal{W}_k\}$, para $k = k_0, k_0+1, k_0+2, \ldots$*

Teorema 3. *El proceso de la carga de un sistema S cualquiera es una cadena de Markov.*

Demostración. La probabilidad de que la carga $\mathcal{W}_k$ al inicio del hiperperiodo k tome un valor dado i, puede expresarse siempre usando probabilidades condicionadas, en función de las probabilidades de $\mathcal{W}_{k-1}$, en la forma siguiente:

$$\mathbb{P}\{\mathcal{W}_k = i\} = \sum_j \mathbb{P}\{\mathcal{W}_{k-1} = j\} \cdot \mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\} \tag{4.12}$$

La anterior ecuación es siempre cierta con independencia de que $\mathcal{W}_k$ sea una cadena de Markov o no. Ahora bien, el término $\mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\}$ representa la probabilidad de que al final del hiperperiodo $k-1$ tengamos una carga de i unidades, *si* al principio era de j unidades. Esta probabilidad se obtiene a partir de la secuencia de activación de los trabajos en el hiperperiodo y de las PFs de sus tiempos de ejecución. Y puesto que en todos los hiperperiodos (a partir del primer hiperperiodo completo) llegan los trabajos en la misma secuencia y con las mismas PFs, estas probabilidades condicionadas serán las mismas en todos los hiperperiodos, e iguales a las del primer hiperperiodo completo. Es decir:

$$\mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\} = \mathbb{P}\{\mathcal{W}_{k_0+1} = i | \mathcal{W}_{k_0} = j\} \quad \text{para } k > k_0$$

Podemos representar esta probabilidad por $b_j(i)$, con lo que la ecuación (4.12) quedará:

$$\mathbb{P}\{\mathcal{W}_k = i\} = \sum_j \mathbb{P}\{\mathcal{W}_{k-1} = j\} \cdot b_j(i) \qquad \text{para } k > k_0 \tag{4.13}$$

Y en esta ecuación se observa que las probabilidades de $\mathcal{W}_k$ dependen tan sólo de las probabilidades de $\mathcal{W}_{k-1}$, y no de las anteriores. Los coeficientes $b_j(i)$ son siempre los mismos e independientes de k (dependen únicamente de los parámetros del sistema). Luego la secuencia $\mathcal{W}_k$ es una cadena de Markov. □

Obtención de la matriz de Markov

La ecuación (4.13) puede escribirse también en forma matricial:

$$\mathbf{b}_k = \mathbf{P}\mathbf{b}_{k-1} \tag{4.14}$$

Siendo $\mathbf{b}_k$ la función de probabilidad de $\mathcal{W}_k$ expresada en forma de vector columna (la fila i-ésima del vector contiene la probabilidad de que $\mathcal{W}_k$ sea igual a i), y siendo $\mathbf{P}$ la matriz de transición definida como:

$$\mathbf{P}(i,j) = b_j(i)$$

Es decir, cada columna $\mathbf{P}(\cdot, j)$ de la matriz $\mathbf{P}$ contiene la PF de la carga al final del primer hiperperiodo completo, *para el caso* en que al principio de dicho hiperperiodo la carga era de j unidades.

La propia definición nos da un mecanismo para obtener los componentes de la matriz **P**. En efecto, para obtener la columna j-ésima de **P**, basta aplicar el algoritmo "convolucionar y encoger" a lo largo del primer hiperperiodo completo, asumiendo una carga inicial de valor j en dicho hiperperiodo. Al llegar al final del mismo, la PF obtenida será la columna j de **P**. Observar que esto implica que cada columna de **P** suma uno, puesto que cada columna es una función de probabilidad.

Cada columna así obtenida tiene un número finito de elementos no nulos, puesto que es el resultado de una serie de "convoluciones y encogimientos" de funciones con un número finito de puntos. Sin embargo, la propia matriz **P** es infinita, pues requiere un número infinito de columnas. Esto plantea una aparente imposibilidad de escribir completamente esta matriz. Sin embargo, se da la circunstancia de que las columnas presentan una cierta regularidad, a partir de una dada, que nos permitirá escribir la *forma general* de la matriz. Usando esta forma, la matriz completa podrá ser especificada usando un número finito de valores.

La forma general de **P**, como demostraremos enseguida, se ajusta al siguiente esquema:

$$\mathbf{P} = \left(\begin{array}{cccc|cccc} & & & & r & & & \\ b_0(0) & b_1(0) & b_2(0) & \dots & b_r(0) & 0 & 0 & 0 \\ b_0(1) & b_1(1) & b_2(1) & \dots & b_r(1) & b_r(0) & 0 & \vdots \\ b_0(2) & b_1(2) & b_2(2) & \dots & b_r(2) & b_r(1) & b_r(0) & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & b_r(2) & b_r(1) & \ddots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & b_r(2) & \ddots \\ b_0(m_r) & b_1(m_r) & b_2(m_r) & \dots & b_r(m_r) & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & \dots & 0 & b_r(m_r) & \vdots & \ddots \\ 0 & 0 & 0 & \dots & 0 & 0 & b_r(m_r) & \ddots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \ddots \end{array}\right) \tag{4.15}$$

Vemos que la matriz puede ser dividida en dos partes. La parte izquierda, sombreada en gris oscuro y formada por las primeras r columnas, desde $\mathbf{P}(\cdot, 0)$ hasta $\mathbf{P}(\cdot, r-1)$ no tiene estructura, es decir, sus coeficientes son en general diferentes entre sí, si bien todas ellas tienen como máximo m_r elementos diferentes de cero[5]. La parte derecha, que comienza en la columna $\mathbf{P}(\cdot, r)$ presenta una regularidad consistente en que los coeficientes se repiten en cada columna, copiados de la columna anterior pero desplazados una unidad hacia abajo. Esta regularidad se

[5] Por supuesto, en estas columnas puede haber ceros también en cualquiera de los elementos $b_j(i)$

ha sombreado en gris claro. El siguiente teorema demuestra que esta estructura de **P** se presenta siempre, sea cual sea el sistema a analizar.

Teorema 4. *Existen dos enteros r y m_r tales que:*

$$\mathbf{P}(i,j) = \begin{cases} b_j(i) & j < r, \quad i \leq m_r \\ \mathbf{P}(i-1, j-1) & j \geq r, \quad i \in [j-r, j-r+m_r] \\ 0 & \textit{para los restantes } i,j \end{cases} \tag{4.16}$$

Demostración. En primer lugar, decir que la expresión (4.16) no es más que otra forma de enunciar matemáticamente la estructura regular de **P** mostrada en la eq. (4.15).

Sea r la máxima cantidad de tiempo que el sistema puede permanecer ocioso en un hiperperiodo cualquiera, a partir del primer hiperperiodo completo. Diremos que r es el "máximo tiempo libre". Esta cantidad puede obtenerse con la fórmula:

$$r = T + W^{\min} - \sum_i \frac{T}{T_i} C_i^{\min} \tag{4.17}$$

donde T es la longitud del hiperperiodo, $W^{\min}$ es la carga existente al final del primer hiperperiodo completo en el caso de que todas las tareas requieran su tiempo mínimo, y $C_i^{\min}$ es el mínimo tiempo de ejecución de la tarea τ_i.

La columna r-ésima de **P** representa la PF de la carga al final de un hiperperiodo completo, para el caso de que al principio la carga fuera de r unidades. Puesto que en este caso la carga inicial es igual al máximo tiempo libre, tenemos garantizado que este hiperperiodo no habrá instantes en que el sistema esté ocioso. Dicho de otro modo, el hiperperiodo completo será un periodo de ocupación (*busy-period*). Es más, siempre que la carga inicial sea mayor de r, también tendremos el hiperperiodo completamente ocupado. A lo largo de un hiperperiodo así, la carga nunca se hace cero, por lo que la carga final en realidad es la suma de la carga inicial más todo el trabajo que se genere en el hiperperiodo, menos la longitud del hiperperiodo. Es decir, si el valor de $\mathcal{W}(k_0 T)$ es r, entonces el de $\mathcal{W}((k_0+1)T)$ será:

$$\mathcal{W}((k_0+1)T) = r - T + \sum_i \frac{T}{T_i} \mathcal{C}_i \tag{4.18}$$

Ahora bien, ya que el sumatorio contiene variables aleatorias, la carga final será también una variable aleatoria. Para obtener su PF es necesario por tanto convolucionar las PFs de todos los trabajos lanzados en el hiperperiodo. El término $r - T$ no es aleatorio, sino determinista, por lo que se traduce en un mero desplazamiento a la derecha de la función resultado.

En resumen, la columna r de la matriz $\mathbf{P}$ se calcula convolucionando las PFs de los tiempos de ejecución de todos los trabajos del hiperperiodo, y desplazando hacia la derecha el resultado $r - T$ unidades. La columna $r + 1$ se calculará de forma análoga, la convolución de todos los trabajos dará el mismo resultado, y la única diferencia es que ahora el desplazamiento final habrá de ser $r + 1 - T$. Y así sucesivamente para todas las columnas a partir de la r. Esto demuestra que todas estas columnas presentan los mismos coeficientes, pero desplazados una posición hacia abajo.

Por otro lado, la columna r tiene un número finito de coeficientes no nulos. Si llamamos al índice del último de ellos m_r, tenemos que m_r representa la carga que habrá al final de un hiperperiodo en el que todos los trabajos requieran su peor tiempo de ejecución, y además la carga inicial sea r (el máximo tiempo libre). Es evidente que si la carga inicial fuese inferior a r, la carga final habrá de ser necesariamente inferior (o igual) a m_r. Esto demuestra que las columnas anteriores a la r han de tener ceros a partir de su fila m_r.

De las dos observaciones anteriores se sigue el enunciado del teorema. □

Por tanto, la matriz completa $\mathbf{P}$ queda especificada a partir de un número finito, $r \times m_r$, de números. Estos números pueden calcularse por el método de "convolucionar y encoger", asumiendo una carga inicial $i = 0, 1, 2, \ldots, r$ y obteniendo la distribución de la carga al final del hiperperiodo.

Estacionariedad y convergencia

Una cadena de Markov representada por una matriz $\mathbf{P}$ puede ser de tipo *recurrente*, si cumple ciertas propiedades que después enunciaremos. Si lo es, está demostrado que ha de existir un vector $\boldsymbol{\pi}$, único excepto por constantes multiplicativas, tal que:

$$\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi} \tag{4.19}$$

Si además la cadena es *positiva*, tal vector $\boldsymbol{\pi}$ es sumable, y por tanto puede normalizarse (dividirse por su suma) y de este modo representar una función de probabilidad. El caso en que no sea positiva y por tanto no sumable, no tendría interés práctico.

Es más, la teoría de procesos estocásticos no sólo demuestra que $\boldsymbol{\pi}$ existe, demuestra también que si la cadena es *recurrente positiva* la función de probabilidad de la cadena de Markov "converge" hacia este vector. Esto es, si la cadena de Markov es $\{\mathcal{X}_i\}$, y la cadena es de tipo positivo recurrente, entonces $f_{\mathcal{X}_i}(\cdot) \to \boldsymbol{\pi}$, cuando $i \to \infty$.

En términos de nuestro análisis esto es muy interesante, ya que significa que si nuestra matriz $\mathbf{P}$ cumpliera las propiedades adecuadas, la cadena sería recurrente positiva, y entonces existiría una *distribución de carga estacionaria*, esto es, una

distribución estadística de la carga tal que, si un hiperperiodo comienza con esa distribución como carga inicial, terminará con esa misma distribución como carga final. Es más, si esta distribución estacionaria existe, el sistema tiende hacia ella, y transcurriendo un número suficiente de hiperperiodos, la PF de la carga al inicio de cada hiperperiodo será prácticamente la misma, e igual a la distribución estacionaria.

Efectivamente, hemos observado este comportamiento en la figura 4.3, para el caso en que $\bar{U} < 1$. Esto tiene una importante implicación práctica, y es que si somos capaces a determinar la distribución estacionaria, podremos analizar el sistema centrándonos en un único hiperperiodo. Basta usar como carga inicial la distribución estacionaria hallada. Este hiperperiodo es representativo de cualquier hiperperiodo en el futuro, una vez alcanzado el régimen permanente. Una vez que el sistema ha estado funcionando por tiempo suficiente, la influencia del transitorio será despreciable, por lo que las PF de los tiempos de respuesta, en promedio, coincidirán con las obtenidas para este hiperperiodo estacionario.

Existencia y unicidad de la distribución estacionaria En esta sección demostraremos rigurosamente que cuando $\bar{U} < 1$, la cadena de Markov es positiva recurrente y de ahí que se observe esa tendencia hacia una distribución estable. Daremos la forma general de ésta distribución y proporcionaremos métodos para computarla numéricamente.

La condición $\bar{U} < 1$ es fácilmente comprensible para el analista, y fácil de verificar dados los parámetros del sistema. En cambio, para demostrar la recurrencia y positividad de la cadena de Markov, sería preferible tener una condición que opere sobre los valores de la matriz de Markov $\mathbf{P}$. En la siguiente proposición encontraremos que decir $\bar{U} < 1$ equivale a decir que la distribución dada por la columna r de $\mathbf{P}$ tiene una media menor que r. Esta condición, aunque suena más extraña, resulta más conveniente para demostrar la recurrencia y positividad de la cadena de Markov, y por eso la necesitamos.

Proposición 3. *Si* $\mathbf{P}(i,j) = b_j(i)$ *es la matriz de transición de la cadena de Markov* $\{\mathcal{W}_k\}$*, que representa la carga al inicio de cada hiperperiodo de un sistema* S *con utilización media* $\bar{U}$*, se cumplen las siguientes equivalencias:*

- $\bar{U} < 1 \iff \sum_i i b_r(i) < r$
- $\bar{U} = 1 \iff \sum_i i b_r(i) = r$
- $\bar{U} > 1 \iff \sum_i i b_r(i) > r$

Demostración. Denotemos por $\bar{B}_r$ al valor esperado de la variable aleatoria cuya distribución viene dada por la columna r de $\mathbf{P}$. Este valor puede calcularse a partir

de los coeficientes de **P**, de acuerdo con la propia definición estadística de valor esperado:

$$\bar{B}_r = \sum_i i\mathbf{P}(i,r) = \sum_i i b_r(i) \tag{4.20}$$

Por otro lado, la columna r es especial, ya que r es el "máximo tiempo libre" del sistema. Un hiperperiodo que comience con una carga inicial de r unidades, estará ocupado durante todo el hiperperiodo incluso en el mejor de los casos. Gracias a esto, la carga final será igual a la suma de todos los tiempos de ejecución, más la carga inicial r menos el tiempo transcurrido T, como se vio en la ecuación (4.18). El valor esperado de esta carga final será la suma de los valores esperados de los términos que la componen. Por tanto también podemos calcular este valor esperado de acuerdo con la ecuación siguiente:

$$\bar{B}_r = r - T + \sum_i \frac{T}{T_i}\bar{C}_i$$

Si sacamos del sumatorio T, por ser constante, el sumatorio que queda representa la utilización media del sistema, tal como se definió en (4.4), por lo que:

$$\bar{B}_r = r - T + T\bar{U} = r + T(\bar{U} - 1)$$

De esta ecuación se deducen las tres equivalencias que enuncia la proposición. □

El siguiente teorema demuestra que la cadena de Markov es recurrente positiva si se cumple que $\bar{U} < 1$. Sin embargo, para esta demostración es necesario añadir la hipótesis adicional de que la cadena sea *irreducible*. Más adelante (pág. 100) explicaremos qué supone esta hipótesis y mostraremos que existen sistemas (bastante comunes) para los que esta hipótesis no se cumple. Sin embargo, mostraremos también que incluso si no se cumple la condición de irreductibilidad, la condición $\bar{U} < 1$ sigue siendo válida para clasificar la cadena como recurrente positiva.

Teorema 5. *Sea un sistema S compuesto por un conjunto de tareas periódicas $\{\tau_i\}$, y sea $\{\mathcal{W}_k\}$ el proceso de la carga para este sistema, que es una cadena de Markov. Si dicha cadena es irreducible, y el sistema cumple $\bar{U} < 1$, entonces la cadena es además recurrente y positiva.*

Demostración. Recientes avances en la teoría de las cadenas de Markov nos dicen [Meyn y Tweedie, 1993; Tweedie, 2000] que, para probar que una cadena de Markov irreducible es además positiva recurrente, basta con probar que se cumplen ciertas *condiciones de deriva* en los coeficientes de su matriz.

En particular, una cadena de Markov irreducible es recurrente si y sólo si existe una función no-negativa $V(x)$, $x \in \mathbb{Z}^+$, tal que $V(x) \to \infty$ cuando $x \to \infty$, y existe un número $N \geq 0$ tal que

$$\sum_j \mathbf{P}(j,x)V(j) \leq V(x) \qquad x > N \tag{4.21}$$

Además de recurrente, la cadena irreducible será positiva si y sólo si existe una función no-negativa $V(x), x \in \mathbb{Z}^+$ y un par de números $N \geq 0, \epsilon > 0$ tales que:

$$\sum_j \mathbf{P}(j,x)V(j) \leq V(x) - \epsilon, \qquad x > N; \tag{4.22}$$

y

$$\sum_j \mathbf{P}(j,x)V(j) < b < \infty, \qquad x \leq N. \tag{4.23}$$

Demostraremos a continuación que la condición $\sum_i i b_r(i) \leq r$ implica que la cadena es recurrente, y más aún, la condición $\sum_i i b_r(i) < r$, implica recurrencia positiva. Puesto que estas condiciones equivalen respectivamente a $\bar{U} \leq 1$ y $\bar{U} < 1$, esto completaría la demostración del teorema.

Para demostrar que la condición (4.21) se cumple, basta elegir como función $V(\cdot)$ y entero N los valores $V(x) = x$, $N = r$. En efecto, debido a la estructura repetitiva de $\mathbf{P}$, a partir de la columna r los coeficientes se repiten, por lo que podemos escribir que:

$$\mathbf{P}(j,x) = b_r(j+r-x) \qquad x > r$$

Si llevamos este hecho, junto con la función $V(x) = x$ y el entero $N = r$, a la condición de deriva (4.21), ésta queda:

$$\sum_j b_r(j+r-x) \cdot j \leq x \qquad x > r$$

Operando en esta inecuación, moviendo x al primer miembro y sumando r a ambos:

$$(r-x) + \sum_j j b_r(j+r-x) \leq r \qquad x > r$$

Por otro lado, al ser $b_r(\cdot)$ una función de probabilidad, su suma es 1, por lo que podemos escribir:

$$(r-x)\sum_j b_r(j+r-x) + \sum_j j b_r(j+r-x) \leq r \qquad x > r$$

Y sacando factor común al sumatorio finalmente queda:

$$\sum_j (j+r-x) b_r(j-x+r) \leq r \qquad x > r$$

Y haciendo un cambio de variable vemos precisamente que la desigualdad a que llegamos es $\sum_i ib_r(i) \leq r$, luego si se cumple esta desigualdad, la cadena es recurrente como queríamos demostrar.

Para demostrar que además es positiva, debemos comprobar que se cumplen también (4.22) y (4.23). Es trivial demostrar, siguiendo los mismos pasos de antes, que la condición (4.22) equivale a $\sum_i ib_r(i) < r$. Por otro lado, la condición (4.23) es equivalente a $\sum_i ib_x(i) < b < \infty$ para todo $x \leq r$. Es decir, esta condición nos pide que el valor esperado para cualquiera de las r primeras columnas esté acotado por un número finito b. Esta condición también se garantiza debido a la estructura especial de nuestra matriz $\mathbf{P}$, que tiene ceros a partir de la fila m_r en sus r primeras columnas como se ve en (4.15).

Por tanto, cuando $\sum_i ib_r(i) < r$ ambas condiciones se cumplen, y la cadena es positiva recurrente. Por otro lado, la proposición 3 demostró que esta condición equivale a $\bar{U} < 1$, lo que completa la demostración del teorema. □

El problema de la irreductibilidad En la terminología de la teoría Markoviana, se dice que un estado i "comunica" con otro estado j, si desde el estado i es posible alcanzar el estado j en un número finito de transiciones. Traduciendo esto al lenguaje de nuestro problema, un valor de la carga inicial i "comunica" con otro valor de la carga inicial j, si al comenzar un hiperperiodo con una carga de i unidades, existe una probabilidad no nula de encontrar en el futuro un hiperperiodo que comience con una carga de j unidades.

Se dice que una cadena de Markov es irreducible cuando todos los estados comunican entre sí. Traduciendo esto al lenguaje de nuestro problema, esta hipótesis supone que cualquier valor de la carga inicial debería ser posible a lo largo de los diferentes hiperperiodos, fuera cual fuese el valor de la carga inicial cuando arrancó el sistema.

Mostraremos un par de contraejemplos en los que se ve que es posible idear sistemas que no cumplen esta hipótesis, y que de hecho no son raros.

Un caso trivial de sistema no irreducible, sería uno que tenga $U^{\min} > 1$. En este sistema, la carga al final de cada hiperperiodo siempre aumenta, por lo que si partimos de una carga inicial de 1 unidad, por ejemplo, nunca podríamos alcanzar en el futuro una carga de 0 unidades. Por tanto el estado 1 no comunica con el estado 0 y la cadena no es irreducible. Sin embargo este caso carece de importancia práctica, puesto que un sistema así sería claramente poco realista.

Un caso menos trivial y más interesante es el siguiente. Considérese un sistema compuesto por una única tarea, cuyos parámetros son:

- *Offset*, $\Phi_1 = 3$,
- Periodo, $T_1 = 4$,

- Tiempo de ejecución, $\mathcal{C}_1$ sólo puede tomar los valores 2 ó 6, ambos con igual probabilidad (pero no ningún valor intermedio).
- Plazo y prioridad no son relevantes para el ejemplo.

La figura 4.17 muestra el patrón de activaciones de este "sistema". La longitud del hiperperiodo evidentemente coincide con el periodo de la única tarea que lo compone. El primer hiperperiodo completo ocurre para $k_0 = 0$.

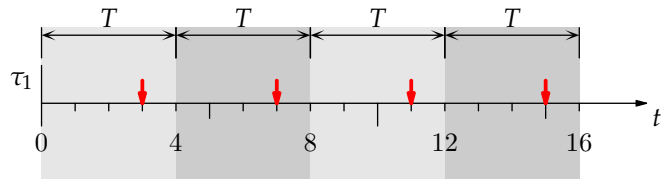


Figura 4.17.: Ejemplo sencillo de un sistema no-irreducible

Vemos que, suponiendo una carga inicial cero, al final del hiperperiodo $(t = 4)$ la carga sólo puede tomar los valores 1 ó 5 (ya que la tarea que se ejecuta en $t = 3$ sólo puede tomar un tiempo de ejecución de 2 ó 6). Desde el estado 0, se pasa al estado 1 ó al 5.

Si la carga inicial es 1, tenemos la misma situación, ya que esa carga inicial se absorbe en el hueco de 3 unidades que hay antes de que llegue la tarea. Por tanto desde el estado 1 sólo se puede pasar de nuevo al estado 1 ó al 5.

Si la carga inicial es 5, cuando llega la tarea aún quedan 2 unidades, que se sumarán al trabajo de la propia tarea, por lo que la carga final puede ser de 3 ó 7. Desde el estado 5 sólo se puede pasar al estado 3 ó al 7.

Finalmente, si la carga inicial es 3, será absorbida antes de que llegue la tarea, por lo que estamos en el mismo caso en que la carga inicial era cero, y la carga final sólo puede tomar los valores 1 ó 5. Desde el estado 3 sólo se puede pasar al estado 1 ó al 5.

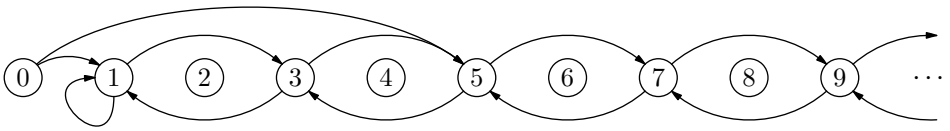


Figura 4.18.: Comunicación entre estados, partiendo del estado cero, para el ejemplo no-irreducible

En la figura 4.18 vemos el patrón general de transiciones partiendo del estado cero. Desde él 0 sólo se puede pasar al estado 1 ó 5, y desde un estado impar sólo se puede alcanzar otro estado impar. Los estados pares no son alcanzables desde

cero, luego la cadena no es reducible. También vemos que si la carga inicial tiene un valor impar, es imposible asimismo que en el futuro tome valor par o cero. Otra razón más de irreductibilidad.

Esta situación no es tan artificial como pueda parecer, puesto que puede darse si los tiempos de ejecución de las tareas tienen valores "prohibidos", es decir, que no pueden ocurrir nunca. Esto puede causar que haya valores prohibidos también para la carga total del sistema, que por tanto sólo podrá evolucionar tomando valores en un cierto subconjunto de los enteros (en el ejemplo anterior, el subconjunto de los impares).

Podríamos analizar también qué ocurre si la carga inicial fuese 4, por ejemplo. Veríamos que en este caso la carga final podría ser 2 ó 6. Parece que con una carga inicial par, sí que se pueden alcanzar valores de carga final pares. Sin embargo obsérvese que una vez alcanzado el valor 2 (el cual se alcanzará eventualmente puesto que la carga tiene una probabilidad no nula de disminuir), quedaremos "atrapados" de nuevo en el subconjunto de los impares. En efecto, si la carga llega a 2, estas 2 unidades serán absorbidas por las 3 unidades de tiempo libre que hay al principio del hiperperiodo siguiente, por lo que la situación será la misma que si la carga inicial hubiese sido cero. Del estado 2 sólo podremos pasar al 1 ó el 5. La figura 4.19 muestra el diagrama de estados completo.

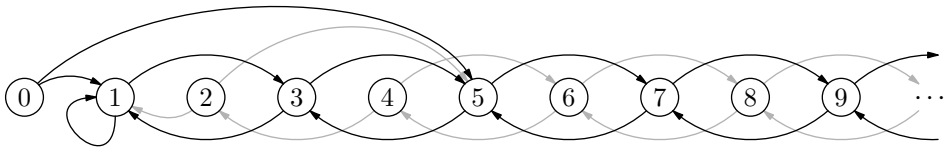


Figura 4.19.: Comunicación entre estados, partiendo de cualquier estado, para el ejemplo no-irreducible

Cabe la pena insistir en que si partimos de un estado par mayor de 2, el sistema se "moverá" durante un tiempo por los estados pares, pero eventualmente "caerá" en el estado 2 y ya no podrá abandonar la secuencia de estados impares. Si el sistema se está ejecutando un tiempo infinito, la fracción de tiempo que ha estado en los estados pares es despreciable frente al total. El comportamiento a largo plazo del sistema está dominado por la cadena moviéndose sólo entre estados impares. Esta idea es crucial.

En general, demostraremos que para cualquier sistema con $\bar{U} < 1$ es posible encontrar un conjunto C de enteros tal que para cualquier par de enteros i, j pertenecientes a C, el estado i comunica con el estado j. En el ejemplo anterior, C sería el conjunto de los números impares. Si construimos una nueva cadena de Markov restringida al conjunto C, esta nueva cadena será irreducible, por lo que se le podrá aplicar el teorema 5. Por otro lado, veremos que el conjunto C es "absorbente", lo que quiere decir que si el estado inicial del sistema original no

pertenece a C, de todas formas está garantizado (probabilidad 1) que al cabo de un número suficiente de transiciones (hiperperiodos) el estado estará en C, por lo que si la cadena restringida era recurrente positiva, la cadena original, sin restringir, también acabará por adoptar una distribución estacionaria, una vez que haya entrado en el conjunto C de estados.

Veamos ahora cómo construir dicho conjunto C. Cabe destacar que esta sección tiene un interés puramente teórico, ya que más adelante daremos un método para obtener la distribución estacionaria $\boldsymbol{\pi}$ que no requiere en absoluto conocer si el sistema es irreducible o no, ni mucho menos obtener de antemano el conjunto C. Esta demostración va encaminada únicamente a convencer al lector de que el hecho de que la cadena original no sea irreducible no tiene importancia práctica, puesto que la existencia de C y su condición "absorbente" garantizan que de todas formas se alcanzará el régimen estacionario también para los sistemas no-irreducibles, con tal de que $\bar{U} < 1$.

Sea $W^{\min}$ el mínimo valor que puede tomar la carga al final de un hiperperiodo. Podemos calcular este valor planteando un hiperperiodo completo en el que todos los trabajos requieran su mínimo tiempo de computación, y asumiendo además que la carga inicial era cero. Por ejemplo, en el sistema de la figura 4.17, el valor de $W^{\min}$ sería 1, que sale cuando la tarea toma su mínimo tiempo posible (2) y la carga inicial es cero.

El valor de $W^{\min}$ así obtenido es un estado que es alcanzable desde cualquier otro. Esto es debido a que, bajo la hipótesis $U^{\min} < 1$, existe una probabilidad no nula de que la carga disminuya en un hiperperiodo, y por tanto existe una probabilidad no nula de que la carga disminuya durante n hiperperiodos consecutivos, hasta alcanzar el valor $W^{\min}$ desde no importa qué valor inicial de la carga. Una vez hallado $W^{\min}$, definimos C como el conjunto de todos los estados alcanzables desde $W^{\min}$. En el ejemplo anterior se trataría del conjunto de los números impares. Está claro que todos los estados contenidos en este C comunican entre sí (siempre se podrá llegar a cualquiera de ellos desde cualquier otro pasando a través del estado $W^{\min}$).

Construyamos ahora una nueva cadena de Markov, que evoluciona únicamente en el espacio de estados definido por C. Para ello, escribimos una nueva matriz, $\mathbf{P}_C$, a partir de la matriz original, $\mathbf{P}$, por el método de tomar de $\mathbf{P}$ solamente los elementos cuyos índices (fila y columna) estén en C. Llamemos $g(\cdot)$ a la función que mapea los índices de $\mathbf{P}_C$ sobre los índices de $\mathbf{P}$. Dicho de otro modo, el elemento $\mathbf{P}_C(i,j)$ es una copia del elemento $\mathbf{P}(g(i),g(j))$.

En el ejemplo de la figura 4.17, tendríamos que C es el conjunto de los números impares; la matriz $\mathbf{P}_C$ se construiría tomando las filas y columnas impares de la matriz $\mathbf{P}$; y la función $g(\cdot)$ se definiría como $g(x) = 2x + 1$.

La cadena de Markov definida por la matriz $\mathbf{P}_C$ es irreducible, puesto que por construcción todos los estados de C comunican unos con otros. Además, es sen-

cillo demostrar que, si el sistema original definido por $\mathbf{P}$ cumplía que $\bar{U} < 1$, entonces la cadena restringida definida por $\mathbf{P}_C$ cumplirá las condiciones de deriva de las ecuaciones (4.22) y (4.23). Para ello basta tomar como función de deriva $V(x) = g(x)$ y como constante N el primer entero tal que $g(N) \geq r$.

De lo anterior se deduce que, si el sistema tiene $\bar{U} < 1$, entonces la cadena de Markov restringida a C es irreducible, positiva y recurrente. Por tanto tendrá una distribución estacionaria $\boldsymbol{\pi}_C$ tal que $\boldsymbol{\pi}_C = \mathbf{P}_C \boldsymbol{\pi}_C$. A partir de este $\boldsymbol{\pi}_C$ podemos definir una distribución estacionaria $\boldsymbol{\pi}$ para la matriz no-restringida original $\mathbf{P}$ en la forma siguiente:

$$\boldsymbol{\pi}(x) = \begin{cases} \boldsymbol{\pi}_C(g^{-1}(x)) & \forall x \in C, \\ 0 & \forall x \notin C. \end{cases} \tag{4.24}$$

En efecto, esta función $\boldsymbol{\pi}(\cdot)$ así definida es una función de probabilidad, puesto que su suma es 1 (ya que la suma de $\boldsymbol{\pi}_C$ era 1). Por otro lado, cuando el sistema tiene $\bar{U} < 1$, la carga "tiende" a disminuir, y a largo plazo existe una probabilidad 1 de que llegue a tomar el valor $W^{\min}$. Puesto que este valor está en C, está garantizado que con el paso del tiempo la cadena de Markov acabará entrando en C, y ya no podrá salir. Por tanto, a largo plazo, la probabilidad de que la cadena esté en un estado fuera de C es cero, ya que un estado así sólo será visitado un número finito de veces, antes de entrar para siempre en C (número infinito de veces). Por esto, definimos $\boldsymbol{\pi}(x)$ como cero, para cualquier x fuera de C. En cambio, si $x \in C$, la probabilidad de estar en el estado x vendrá dada por $\boldsymbol{\pi}_C(\cdot)$, tras "deshacer" el cambio de índices a través de la función $g(\cdot)$.

Si aplicamos estas ideas al ejemplo de la figura 4.17, tendríamos que la distribución estacionaria de la carga para ese sistema se calcularía en la forma siguiente:

1. Construir la matriz $\mathbf{P}_C$ tomando de $\mathbf{P}$ sólo los elementos con ambos índices impares.

2. Obtener la distribución estacionaria $\boldsymbol{\pi}_C$ de la matriz $\mathbf{P}_C$ (en la siguiente sección abordaremos este problema)

3. Construir la distribución estacionaria $\boldsymbol{\pi}(x)$ a partir de la $\boldsymbol{\pi}_C(\cdot)$ antes obtenida, sin más que asignar cero para todo punto con x par, y asignar $\boldsymbol{\pi}_C((x-1)/2)$ a todo punto con x impar.

En realidad, lo anterior no es necesario en la práctica, y se expone aquí por un interés puramente teórico. Si directamente obtenemos la distribución estacionaria de la matriz $\mathbf{P}$ por el método que veremos enseguida, olvidándonos de si la cadena de Markov era irreducible o no, hallaremos una distribución $\boldsymbol{\pi}$ que automáticamente tendrá ceros en los lugares correctos (correspondientes a índices fuera de C). El sólo hecho de que el sistema tenga $\bar{U} < 1$ garantiza que existe una

distribución estacionaria única, por lo que podemos buscarla directamente sin necesidad de restringir antes la cadena al conjunto C. Esto es muy conveniente porque, en general, determinar el conjunto C puede ser extremadamente difícil.

Cálculo de la distribución estacionaria

Hemos demostrado en la sección precedente que, si el sistema tiene $\bar{U} < 1$, entonces existe un único vector $\boldsymbol{\pi}$ que sume 1 y tal que:

$$\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$$

Ahora la pregunta es ¿podemos conocer el valor de las componentes de éste vector? La ecuación anterior parece plantear que $\boldsymbol{\pi}$ sería un vector propio de la matriz $\mathbf{P}$, correspondiente al valor propio 1. Si $\mathbf{P}$ fuera finita, digamos de dimensión $N \times N$, entonces efectivamente la ecuación matricial anterior daría origen a un conjunto de N ecuaciones con N incógnitas (que serían las N componentes de $\boldsymbol{\pi}$), y bastaría solucionar este sistema de ecuaciones para obtener la respuesta buscada.

Por desgracia la matriz $\mathbf{P}$ es infinita, por lo que si tratamos de desarrollar la ecuación matricial anterior, obtendremos un sistema de infinitas ecuaciones con infinitas incógnitas que claramente no puede ser resuelto.

Presentaré en esta sección un método que permite obtener los infinitos coeficientes de $\boldsymbol{\pi}$. El método, como se verá, proporciona de forma numérica los primeros elementos, junto con una ecuación en forma cerrada que permite obtener cualquiera de los restantes. El interés de esta ecuación es que muestra la forma general que tiene la distribución $\boldsymbol{\pi}$, para cualquier sistema. Además, si el sistema es muy simple puede aplicarse este método para obtener de forma exacta tantos elementos de $\boldsymbol{\pi}$ como se desee. En cambio, para sistemas grandes, la complejidad del problema (número de ecuaciones a resolver) hace este método inaplicable, y deberemos recurrir a aproximaciones que se tratarán en la sección 4.6.

En lugar de dar una descripción formal del método, que sería muy farragosa, daré un ejemplo que se resolverá paso a paso, mencionando donde sea necesario cómo se generalizaría el ejemplo a cualquier otro caso. El sistema de ejemplo se muestra en el cuadro 4.7.

Este sistema tiene un hiperperiodo de longitud 12, con el patrón de activaciones de trabajos que se muestra en la figura 4.20.

Ya que las tareas tienen *offset* cero, el primer hiperperiodo completo empieza en $t = 0$. El sistema tiene una $U^{\max}$ de $16/12$, que es mayor de uno, por lo que la carga al final del primer hiperperiodo tiene diferente distribución que la carga inicial. No se puede analizar el primer hiperperiodo y extrapolar a los demás. Por otro lado, la $\bar{U}$ del sistema es 0.925, inferior a 1, por lo que de acuerdo con el teorema 5 el sistema tendrá una distribución estacionaria única tal que $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$.

Tarea	*Offset*	Periodo	Prioridad	Tiempo de ejecución ($\mathcal{C}_i$)
τ_1	0	4	alta	$\{1,2\}$ con la misma probabilidad
τ_2	0	6	baja	$\{2,3,4\}$ con prob. 0.2, 0.3 y 0.5, respect.

Cuadro 4.7.: Sistema de ejemplo para obtención de la distribución estacionaria de la carga

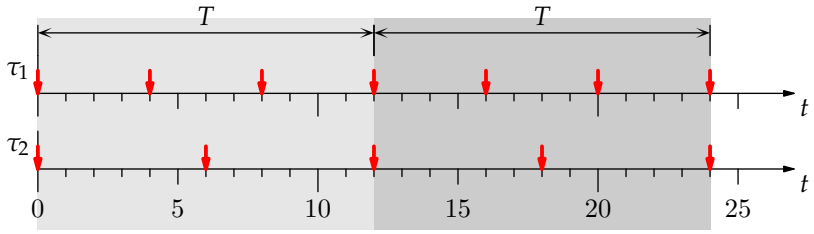


Figura 4.20.: Patrón de activaciones del ejemplo

En primer lugar debemos encontrar la matriz **P**. Para ello, basta aplicar el método "convolucionar y encoger" a lo largo del primer hiperperiodo, partiendo en el instante $t = 0$ hasta llegar al instante $t = 12$, y usando como carga inicial una carga determinista de valor k. La PF de la carga para el instante 12 así obtenida, será la columna k de la matriz **P**. De acuerdo con el teorema 4, la matriz **P** tiene una estructura tal que sólo es necesario calcular $r+1$ columnas, pues el resto se repiten. El valor de r fue definido en la ecuación (4.17) que repetimos aquí para comodidad del lector:

$$r = T + W^{\min} - \sum_i \frac{T}{T_i} C_i^{\min}$$

En este ejemplo, por lo que $r = 12 + 0 - (3 \times 1 + 2 \times 2) = 5$, de modo que sólo necesitamos calcular las columnas 0 a 5 de **P**. El resultado es el siguiente (los guiones representan el cero):

$$\mathbf{P} = \left(\begin{array}{ccccc|cc|c} & & & & & & r & \\ 0{,}8375 & 0{,}595 & 0{,}3275 & 0{,}13125 & 0{,}035 & 0{,}005 & - & \dots \\ 0{,}13125 & 0{,}2425 & 0{,}2675 & 0{,}19625 & 0{,}09625 & 0{,}03 & 0{,}005 & \ddots \\ 0{,}03125 & 0{,}13125 & 0{,}2425 & 0{,}2675 & 0{,}19625 & 0{,}09625 & 0{,}03 & \ddots \\ - & 0{,}03125 & 0{,}13125 & 0{,}2425 & 0{,}2675 & 0{,}19625 & 0{,}09625 & \ddots \\ - & - & 0{,}03125 & 0{,}13125 & 0{,}2425 & 0{,}2675 & 0{,}19625 & \ddots \\ - & - & - & 0{,}03125 & 0{,}13125 & 0{,}2425 & 0{,}2675 & \ddots \\ - & - & - & - & 0{,}03125 & 0{,}13125 & 0{,}2425 & \ddots \\ - & - & - & - & - & 0{,}03125 & 0{,}13125 & \ddots \\ - & - & - & - & - & - & 0{,}03125 & \ddots \\ - & - & - & - & - & - & - & \ddots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{array}\right) \tag{4.25}$$

Hemos calculado también la columna 6 para verificar que, efectivamente, es idéntica a la columna 5, sólo que desplazada una posición hacia abajo. La columna 5 tiene 8 elementos distintos de cero, por lo que $m_r = 7$ en este ejemplo. En general m_r será igual a $r - T + \sum_i (T/T_i) C_i^{\max}$. Comprobamos que, como estaba previsto, ninguna de las cinco primeras columnas tiene más de $m_r + 1$ elementos no nulos.

Observemos otro hecho fundamental. Debido a que las columnas se repiten desplazadas a partir de la columna r, también ocurre que las filas se repiten desplazadas a partir de la fila m_r. En realidad, en este ejemplo particular, este patrón a nivel de fila aparece mucho antes, en la fila 2 (comenzando a numerarlas por cero), pero en el caso general esta regularidad sólo se puede garantizar a partir de la fila m_r.

Se trata pues de resolver la ecuación $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$ siendo $\mathbf{P}$ la matriz anterior. Si desarrollamos la ecuación matricial en un sistema de ecuaciones, obtendremos infinitas ecuaciones, e infinitas incógnitas que llamaremos $\pi_0, \pi_1, \dots$ que son los componentes de $\boldsymbol{\pi}$. Sin embargo, sólo las primeras $m_r + 1$ ecuaciones son "realmente diferentes", ya que debido a la regularidad a nivel de fila, las ecuaciones a partir de la m_r tienen siempre los mismos coeficientes, sólo que afectando a diferentes incógnitas.

Podemos dividir por tanto el sistema de infinitas ecuaciones en dos partes. En primer lugar tenemos un conjunto de $(m_r + 1)$ ecuaciones con $(m_r + r + 1)$ incógnitas. En nuestro ejemplo son 8 ecuaciones con 13 incógnitas, que son las si-

guientes:

$$\begin{aligned}
\pi_0 &= 0{,}8375\pi_0 + 0{,}595\pi_1 + 0{,}3275\pi_2 + 0{,}13125\pi_3 + 0{,}035\pi_4 + 0{,}005\pi_5 \\
\pi_1 &= 0{,}13125\pi_0 + 0{,}2425\pi_1 + 0{,}2675\pi_2 + 0{,}19625\pi_3 + 0{,}09625\pi_4 \\
&\quad + 0{,}03\pi_5 + 0{,}005\pi_6 \\
\pi_2 &= 0{,}03125\pi_0 + 0{,}13125\pi_1 + 0{,}2425\pi_2 + 0{,}2675\pi_3 + 0{,}19625\pi_4 \\
&\quad + 0{,}09625\pi_5 + 0{,}03\pi_6 \\
&\vdots \\
\pi_7 &= 0{,}03125\pi_5 + 0{,}13125\pi_6 + 0{,}2425\pi_7 + 0{,}2675\pi_8 + 0{,}19625\pi_9 + 0{,}09625\pi_{10} \\
&\quad + 0{,}03\pi_{11} + 0{,}005\pi_{12}
\end{aligned} \tag{4.26}$$

Y por otro lado, las restantes ecuaciones tienen todas una misma forma general, debido a que los coeficientes se van repitiendo desplazados hacia la derecha. Podemos resumir las restantes ecuaciones en una sola expresión, válida para $j > m_r + 1$:

$$\begin{aligned}
\pi_j = {} & 0{,}03125\pi_{j-2} + 0{,}13125\pi_{j-1} + 0{,}24250\pi_j + {+}0{,}26750\pi_{j+1} + 0{,}19625\pi_{j+2} \\
& + 0{,}09625\pi_{j+3} + 0{,}03000\pi_{j+4} + 0{,}00500\pi_{j+5}
\end{aligned}$$

Observar que los coeficientes en esta ecuación son los mismos que los que aparecen en la columna r de $\mathbf{P}$, y que como ya se explicó estos coeficientes son el resultado de la convolución de las PFs de los tiempos de ejecución de todos los trabajos lanzados en un hiperperiodo completo.

Despejando el último término (π_{j+5}) obtenemos la ecuación:

$$\begin{aligned}
\pi_{j+5} = {} & -6{,}25\pi_{j-2} - 26{,}25\pi_{j-1} + 151{,}5\pi_j - 53{,}5\pi_{j+1} \\
& - 39{,}25\pi_{j+2} - 19{,}25\pi_{j+3} - 6\pi_{j+4}
\end{aligned} \tag{4.27}$$

Esta ecuación es una relación de recurrencia, que es cierta para todo $j > m_r + 1$, en nuestro ejemplo, para $j > 8$. Para $j = 9$, la ecuación anterior nos permitiría calcular π_{14} a partir de $\pi_{13}, \pi_{12}, \ldots \pi_7$, si conociéramos los valores de estos primeros componentes. Y aplicando de nuevo la misma ecuación para $j = 10$, obtendríamos π_{15} a partir de los 7 componentes anteriores, etc. Por tanto, si pudiéramos determinar los primeros $m_r + r + 1$ componentes de $\boldsymbol{\pi}$, usando la ecuación (4.27) repetidamente podríamos determinar los siguientes. Por desgracia, nos faltan ecuaciones para poder determinar los primeros componentes de $\boldsymbol{\pi}$.

Pero podemos hacer uso de un hecho, y es que sabemos que $\bar{U} < 1$ y que por tanto $\boldsymbol{\pi}$ existe y es único. La existencia de $\boldsymbol{\pi}$ nos servirá para plantear nuevas ecuaciones que permitirán resolver el problema.

Comenzamos por reescribir la ecuación (4.27) en forma matricial:

$$\begin{pmatrix} \pi_{j-1} \\ \pi_j \\ \pi_{j+1} \\ \pi_{j+2} \\ \pi_{j+3} \\ \pi_{j+4} \\ \pi_{j+5} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -6{,}25 & -26{,}25 & 151{,}5 & -53{,}5 & -39{,}25 & -19{,}25 & -6 \end{pmatrix} \begin{pmatrix} \pi_{j-2} \\ \pi_{j-1} \\ \pi_j \\ \pi_{j+1} \\ \pi_{j+2} \\ \pi_{j+3} \\ \pi_{j+4} \end{pmatrix} \tag{4.28}$$

Y si llamamos $\mathbf{Q}_j$ al vector columna $(\pi_{j-2}, \pi_{j-1}, \pi_j, \pi_{j+1}, \pi_{j+2}, \pi_{j+3}, \pi_{j+4})^\mathsf{T}$, y $\mathbf{A}$ a la matriz de coeficientes, la recurrencia se expresa más concisamente como $\mathbf{Q}_{j+1} = \mathbf{A}\,\mathbf{Q}_j$ y es válida para $j > 8$. Observar que la dimensión de la matriz $\mathbf{A}$ es m_r, y que la última fila de la misma se obtiene trivialmente a partir de los coeficientes de la columna r de $\mathbf{P}$.

La forma matricial nos permite ver fácilmente que $\mathbf{Q}_9 = \mathbf{A}\mathbf{Q}_8$, $\mathbf{Q}_{10} = \mathbf{A}^2\mathbf{Q}_8$, $\mathbf{Q}_{11} = \mathbf{A}^3\mathbf{Q}_8$, etc. En general:

$$\mathbf{Q}_n = \mathbf{A}^{n-8}\mathbf{Q}_8 \qquad \text{para } n > 8 \tag{4.29}$$

Para poder calcular fácilmente $\mathbf{A}^{n-8}$, diagonalizaremos esta matriz. Es decir, reescribimos $\mathbf{A}$ en la forma $\mathbf{A} = \mathbf{V}^{-1}\mathbf{D}\mathbf{V}$, donde $\mathbf{V}^{-1}$ es la matriz cuyas columnas $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_7$ son los vectores propios de $\mathbf{A}$; la matriz $\mathbf{D}$ es una matriz diagonal cuyos elementos son $\lambda_1, \lambda_2, \ldots, \lambda_7$, los valores propios de $\mathbf{A}$; y la matriz $\mathbf{V}$ es la inversa de $\mathbf{V}^{-1}$. Una vez realizada la diagonalización, elevar $\mathbf{A}$ a cualquier potencia es mucho más sencillo, ya que basta elevar a esa potencia la matriz diagonal $\mathbf{D}$, es decir, $\mathbf{A}^x = \mathbf{V}^{-1}\mathbf{D}^x\mathbf{V}$. Esto nos lleva a la siguiente ecuación:

$$\begin{aligned} \mathbf{Q}_n &= \mathbf{V}^{-1}\mathbf{D}^{n-8}\mathbf{V}\,\mathbf{Q}_8 \\ &= c_1\lambda_1^{n-8}\mathbf{v}_1 + c_2\lambda_2^{n-8}\mathbf{v}_2 + c_3\lambda_3^{n-8}\mathbf{v}_3 + c_4\lambda_4^{n-8}\mathbf{v}_4 \\ &\quad + c_5\lambda_5^{n-8}\mathbf{v}_5 + c_6\lambda_6^{n-8}\mathbf{v}_6 + c_7\lambda_7^{n-8}\mathbf{v}_7 \end{aligned} \tag{4.30}$$

donde los términos c_i son unos coeficientes reales que vienen dados por la ecuación

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7)^\mathsf{T} = \mathbf{V} \cdot \mathbf{Q}_8 = \mathbf{V} \cdot (\pi_6, \pi_7, \pi_8, \pi_9, \pi_{10}, \pi_{11}, \pi_{12})^\mathsf{T} \tag{4.31}$$

Para hallar los valores propios de **A** es necesario resolver su polinomio característico. Éste puede encontrarse de forma muy simple, sin que sea necesario siquiera escribir la matriz **A**, directamente a partir de los coeficientes de la columna r de **P**. Puede demostrarse fácilmente que la forma general de este polinomio es:

$$f(\lambda) = \Big(\sum_{i=0}^{m_r} b_r(i)\lambda^{m_r-i} \Big) - \lambda^{m_r-r} = 0 \tag{4.32}$$

Observar que el grado del polinomio es m_r. En el ejemplo, tras plantear y resolver el polinomio característico se obtienen las siguientes raíces:

$$\begin{aligned}
\lambda_1 &= (-3{,}2976 + 1{,}825i) \\
\lambda_2 &= (-3{,}2976 - 1{,}825i) \\
\lambda_3 &= (-0{,}3099 + 3{,}0755i) \\
\lambda_4 &= (-0{,}3099 - 3{,}0755i) \\
\lambda_5 &= 1 \\
\lambda_6 &= 0{,}3476 \\
\lambda_7 &= -0{,}1325
\end{aligned}$$

Entre ellas siempre habrá una solución de valor 1, puesto que todas las filas de **A** siempre suman 1. Entre las soluciones restantes, habrá algunas que tengan módulo mayor de 1 y otras que tengan módulo menor de 1. Sin embargo, si miramos la ecuación (4.30), vemos que esto plantea una aparente paradoja. En efecto, al ir todos los λ_i elevados a $(n-8)$, si algunos de ellos son mayores de 1, los términos correspondientes tenderán a infinito a medida que crece n, mientras que los términos con $\lambda_i < 1$ irán tendiendo a cero. Los términos que tienden a cero no son problema, pero los que tienden a infinito sí, porque hacen que los diferentes elementos de la solución, los valores π_i, también tiendan a infinito. Si esto ocurriera, $\boldsymbol{\pi}$ no sería sumable, y por tanto la distribución estacionaria no existiría.

Pero $\boldsymbol{\pi}$ existe, lo hemos demostrado antes. Por tanto esta paradoja sólo puede resolverse si los términos con $\lambda_i \geq 1$ desaparecen de la ecuación (4.30). Es decir, si los correspondientes c_i son cero. Puesto que los c_i aún no han sido determinados (son incógnitas del problema), podemos forzar a que algunos de ellos sean cero. De este modo obtenemos nuevas ecuaciones. En este ejemplo particular, los valores propios con módulo mayor de 1 son $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ y λ_5, por lo que, para garantizar la existencia de $\boldsymbol{\pi}$, los coeficientes c_1, c_2, c_3, c_4 y c_5 han de ser cero.

Esto dará origen a 5 nuevas ecuaciones, que plantearemos a continuación. Para ello necesitamos conocer los valores de la matriz **V**, lo que implica a su vez obtener los vectores propios de **A**. En el ejemplo, los resultados son:

$$\mathbf{v}_1 = \begin{pmatrix} -0{,}00033 + 0{,}00004i \\ +0{,}00104 - 0{,}00073i \\ -0{,}00208 + 0{,}00430i \\ -0{,}00098 - 0{,}01798i \\ +0{,}03604 + 0{,}05751i \\ -0{,}22382 - 0{,}12387i \\ +0{,}96416 + 0{,}00000i \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} -0{,}00033 - 0{,}00004i \\ +0{,}00104 + 0{,}00073i \\ -0{,}00208 - 0{,}00430i \\ -0{,}00098 + 0{,}01798i \\ +0{,}03604 - 0{,}05751i \\ -0{,}22382 + 0{,}12387i \\ +0{,}96416 + 0{,}00000i \end{pmatrix}$$

$$\mathbf{v}_3 = \begin{pmatrix} +0{,}00089 - 0{,}00061i \\ +0{,}00161 + 0{,}00294i \\ -0{,}00954 + 0{,}00405i \\ -0{,}00951 - 0{,}03059i \\ +0{,}09704 - 0{,}01976i \\ +0{,}03069 + 0{,}30457i \\ -0{,}94623 + 0{,}00000i \end{pmatrix} \qquad \mathbf{v}_4 = \begin{pmatrix} +0{,}00089 + 0{,}00061i \\ +0{,}00161 - 0{,}00294i \\ -0{,}00954 - 0{,}00405i \\ -0{,}00951 + 0{,}03059i \\ +0{,}09704 + 0{,}01976i \\ +0{,}03069 - 0{,}30457i \\ -0{,}94623 + 0{,}00000i \end{pmatrix}$$

$$\mathbf{v}_5 = \begin{pmatrix} -0{,}37796 \\ -0{,}37796 \\ -0{,}37796 \\ -0{,}37796 \\ -0{,}37796 \\ -0{,}37796 \\ -0{,}37796 \end{pmatrix} \qquad \mathbf{v}_6 = \begin{pmatrix} -0{,}9376558 \\ -0{,}3258970 \\ -0{,}1132706 \\ -0{,}0393690 \\ -0{,}0136833 \\ -0{,}0047558 \\ -0{,}0016530 \end{pmatrix} \qquad \mathbf{v}_7 = \begin{pmatrix} -0{,}99118 \\ +0{,}13132 \\ -0{,}017398 \\ +0{,}002305 \\ -0{,}000305 \\ +0{,}000040 \\ -0{,}000005 \end{pmatrix}$$

Estos vectores nos dan directamente las columnas de la matriz $\mathbf{V}^{-1}$, y hallando su inversa obtenemos la matriz $\mathbf{V}$ (se redondea al tercer decimal para ahorrar espacio):

$$\mathbf{V} = \begin{pmatrix} 0{,}512 + 0{,}911i & 2{,}147 + 3{,}549i & -12{,}443 - 23{,}182i & 4{,}289 + 14{,}778i & 4{,}116 + 1{,}739i & 0{,}844 + 1{,}873i & 0{,}536 + 0{,}331i \\ 0{,}512 - 0{,}911i & 2{,}147 - 3{,}549i & -12{,}443 + 23{,}182i & 4{,}289 - 14{,}778i & 4{,}116 - 1{,}739i & 0{,}844 - 1{,}873i & 0{,}536 - 0{,}331i \\ 0{,}503 - 0{,}009i & 2{,}094 - 0{,}199i & -12{,}331 - 0{,}453i & 4{,}562 + 3{,}908i & 4{,}270 - 1{,}651i & 0{,}880 - 1{,}348i & 0{,}021 - 0{,}248i \\ 0{,}503 + 0{,}009i & 2{,}094 + 0{,}199i & -12{,}331 + 0{,}453i & 4{,}562 - 3{,}908i & 4{,}270 + 1{,}651i & 0{,}880 + 1{,}348i & 0{,}021 + 0{,}248i \\ +0{,}092 & +0{,}478 & -1{,}749 & -0{,}963 & -0{,}386 & -0{,}103 & -0{,}015 \\ -0{,}373 & -2{,}636 & +1{,}445 & +0{,}968 & +0{,}445 & +0{,}131 & +0{,}021 \\ -0{,}691 & +2{,}314 & -0{,}712 & -0{,}539 & -0{,}271 & -0{,}086 & -0{,}015 \end{pmatrix}$$

Una vez conocida $\mathbf{V}$, la ecuación matricial (4.31) puede ser reescrita en forma

de sistema de ecuaciones:

$$\begin{aligned}
c_1 &= (0{,}512 + 0{,}911i)\pi_6 + (2{,}147 + 3{,}549i)\pi_7 + (-12{,}443 - 23{,}182i)\pi_8 \\
&\quad + (4{,}289 + 14{,}778i)\pi_9 + (4{,}116 + 1{,}739i)\pi_{10} + (0{,}844 + 1{,}873i)\pi_{11} \\
&\quad + (0{,}536 + 0{,}331i)\pi_{12} \\
c_2 &= (0{,}512 - 0{,}911i)\pi_6 + (2{,}147 - 3{,}549i)\pi_7 + (-12{,}443 + 23{,}182i)\pi_8 \\
&\quad + (4{,}289 - 14{,}778i)\pi_9 + (4{,}116 - 1{,}739i)\pi_{10} + (0{,}844 - 1{,}873i)\pi_{11} \\
&\quad + (0{,}536 - 0{,}331i)\pi_{12} \\
&\vdots \\
c_5 &= 0{,}092\pi_6 + 0{,}478\pi_7 - 1{,}749\pi_8 - 0{,}963\pi_9 - 0{,}386\pi_{10} - 0{,}103\pi_{11} - 0{,}015\pi_{12} \\
c_6 &= -0{,}373\pi_6 - 2{,}636\pi_7 + 1{,}445\pi_8 + 0{,}968\pi_9 + 0{,}445\pi_{10} + 0{,}131\pi_{11} + 0{,}021\pi_{12} \\
c_7 &= -0{,}691\pi_6 + 2{,}314\pi_7 - 0{,}712\pi_8 - 0{,}539\pi_9 - 0{,}271\pi_{10} - 0{,}086\pi_{11} - 0{,}015\pi_{12}
\end{aligned} \tag{4.33}$$

Como hemos dicho, para que la solución $\boldsymbol{\pi}$ exista es necesario que los coeficientes c_1, c_2, c_3, c_4 y c_5 sean cero, por lo que llevando este dato a las ecuaciones anteriores, tenemos que las cinco primeras se convierten en restricciones adicionales que deben cumplir las primeras componentes de $\boldsymbol{\pi}$. Si sumamos estas 5 nuevas ecuaciones a las 8 que teníamos inicialmente (obtenidas de las 8 primeras filas de la matriz de Markov $\mathbf{P}$), tendremos en total 13 ecuaciones y 13 incógnitas $(\pi_0, \ldots, \pi_{12})$, luego parece que el sistema podría ser resuelto.

Sin embargo esto no es así, ya que una de las cinco ecuaciones que acabamos de obtener es linealmente dependiente de las demás. Se trata de la ecuación resultante de hacer $c_5 = 0$, y esto es así porque c_5 es el coeficiente que afecta al valor propio λ_5, que es el de valor 1 que ya hemos dicho que aparece siempre y no aporta información adicional. De modo que tenemos en realidad 12 ecuaciones linealmente independientes y 13 incógnitas.

Se puede demostrar que esto siempre ocurrirá para cualquier sistema, es decir, que finalizaremos con un conjunto de $m_r + r + 1$ ecuaciones, de las que una debe ser descartada, y $m_r + r + 1$ incógnitas. Estas ecuaciones provienen de dos planteamientos diferentes: las $(m_r + 1)$ primeras provienen directamente de las primeras filas de $\mathbf{P}$; las r restantes provienen de plantear que algunos de los c_i han de ser cero en la ecuación (4.31). Para esta segunda parte es necesario diagonalizar la matriz $\mathbf{A}$. Para obtener r nuevas ecuaciones, tiene que ocurrir que al hallar los valores propios de $\mathbf{A}$, aparezcan siempre r valores propios con módulo mayor o igual a uno. Es decir, que el polinomio (4.32) debe tener siempre r soluciones fuera del disco unidad. Se puede demostrar que esto siempre ocurrirá con tal de que $\bar{U} < 1$. La demostración puede encontrarse en el apéndice B.1.

De todas formas, aún nos falta una ecuación para poder determinar las primeras componentes de $\boldsymbol{\pi}$. Por tanto, de momento, lo que tenemos es un conjunto de

ecuaciones en las que podemos poner todo en función de la primera componente π_0. Como son ecuaciones lineales, lo que tenemos son infinitas soluciones $\boldsymbol{\pi}$, que en realidad son todas la misma excepto por un factor multiplicativo. De todas estas soluciones, sólo una tendrá su suma igual a 1. La ecuación que falta que nos permitirá hallar $\boldsymbol{\pi}$ es por tanto:

$$\sum_{i=0}^{\infty} \pi_i = 1$$

Una forma de resolverlo puede ser simplemente hacer $\pi_0 = 1$, resolver el sistema de $(m_r + r)$ ecuaciones que ahora tiene $(m_r + r)$ incógnitas, y de esta forma obtener las primeras $(m_r + r)$ componentes de $\boldsymbol{\pi}$. Una vez obtenidas, llevamos estas soluciones al sistema (4.33) de donde podremos obtener los valores de los c_i que eran distintos de cero (en nuestro ejemplo obtendríamos así los valores de c_6 y c_7). Finalmente, llevaremos estos c_i a la ecuación (4.30), junto con los valore propios λ_i de módulo menor de 1, y obtendremos de esta forma la fórmula general que nos da el valor de π_n para cualquier $n > m_r$, y habremos determinado por consiguiente la forma completa de $\boldsymbol{\pi}$.

Sigamos estos pasos en nuestro ejemplo:

1. Hacer $\pi_0 = 1$ y resolver el sistema de 12 ecuaciones formado por:

   - Las 8 ecuaciones mostradas en (4.26)
   - Las 4 ecuaciones obtenidas al hacer $c_1 = c_2 = c_3 = c_4 = 0$ en (4.33). Observar que algunas de las ecuaciones en (4.33) incluyen números complejos, por ejemplo la ecuación que fuerza $c_1 = 0$. Ya que hay que forzar que tanto la parte real como la imaginaria sean cero, cada una de estas ecuaciones daría lugar a dos. Pero por otro lado, para cada ecuación con números complejos existe otra con los mismos números conjugados. Así, en el ejemplo, la conjugada de $c_1 = 0$ es $c_2 = 0$. Puesto que la forma conjugada da lugar a las mismas ecuaciones, al final el número de ecuaciones total se mantiene en 4, y no se dobla como pudiera parecer a primera vista.

   Tras resolver el sistema obtenemos:

   $$\begin{array}{lll} \pi_1 = 0{,}21508 & \pi_5 = 0{,}003661 & \pi_9 = 0{,}000053 \\ \pi_2 = 0{,}09230 & \pi_6 = 0{,}001278 & \pi_{10} = 0{,}000019 \\ \pi_3 = 0{,}02976 & \pi_7 = 0{,}000443 & \pi_{11} = 6{,}4737\,10^{-6} \\ \pi_4 = 0{,}01065 & \pi_8 = 0{,}000154 & \pi_{12} = 2{,}2500\,10^{-6} \end{array}$$

2. Llevar los resultados al sistema (4.33) para determinar los valores de c_6 y c_7 (los demás deben ser cero, lo que también puede verificarse en este punto).

En nuestro caso obtenemos:

$$c_6 = -0{,}00136122 \qquad c_7 = -1{,}50568\,10^{-6}$$

3. Llevar estos c_6 y c_7, junto con los valores propios $\lambda_6 = 0{,}3476$ y $\lambda_7 = -0{,}1325$ y los vectores propios $\mathbf{v}_6$ y $\mathbf{v}_7$, a la ecuación (4.30), para obtener:

$$\begin{pmatrix} \pi_{n-2} \\ \pi_{n-1} \\ \pi_n \\ \pi_{n+1} \\ \pi_{n+2} \\ \pi_{n+3} \\ \pi_{n+4} \end{pmatrix} = -0{,}00136 \begin{pmatrix} -0{,}93766 \\ -0{,}32590 \\ -0{,}11327 \\ -0{,}03937 \\ -0{,}01368 \\ -0{,}00476 \\ -0{,}00165 \end{pmatrix} (0{,}3476)^{n-8} - 1{,}5057\,10^{-6} \begin{pmatrix} -0{,}99118 \\ +0{,}13132 \\ -0{,}01740 \\ +0{,}00231 \\ -0{,}00031 \\ +0{,}00004 \\ -0{,}00000 \end{pmatrix} (-0{,}1325)^{n-8}$$

La ecuación matricial anterior es válida, como vimos, para $n > 8$, y da lugar a 7 ecuaciones que dan la forma general de $\boldsymbol{\pi}$. En realidad, las 7 ecuaciones describen la misma forma, por lo que podemos tomar una cualquiera de ellas, por ejemplo la primera:

$$\pi_{n-2} = (-0{,}0136)(-0{,}93766)(0{,}3476)^{n-8} - (1{,}5057\,10^{-6})(-0{,}99118)(-0{,}1325)^{n-8}$$

válida para $n > 8$. Cambiamos $n-2$ por n, y operamos para obtener:

$$\pi_n = 0{,}0127635(0{,}3476)^{n-6} + 1{,}49242\,10^{-6}(-0{,}1325)^{n-6} \quad n > 6 \tag{4.34}$$

Por tanto tenemos ya una expresión que nos da todas las componentes de $\boldsymbol{\pi}$ a partir de la sexta (las seis primeras ya las teníamos del sistema de ecuaciones).

Sin embargo, para poder resolver el sistema hemos hecho la hipótesis de que $\pi_0 = 1$, lo cual es claramente falso, puesto que en este caso la suma de todos los componentes de $\boldsymbol{\pi}$ no puede ser 1. El último paso será entonces calcular la suma del $\boldsymbol{\pi}$ que hemos obtenido y dividir todos sus términos por ella, para obtener un nuevo $\boldsymbol{\pi}$ que pueda ser una función de probabilidad y que por tanto será la distribución de la carga estacionaria que buscábamos.

La suma de los componentes de $\boldsymbol{\pi}$ se puede hallar, a pesar de que $\boldsymbol{\pi}$ tiene infinitos componentes, gracias a que a partir de un componente dado todos los demás siguen la ecuación (4.34) que es una suma de exponenciales. Por tanto:

$$\begin{aligned}\sum_{i=0}^{\infty}\pi_i &= \sum_{i=0}^{6}\pi_i + \sum_{i=7}^{\infty}\pi_i \\ &= \sum_{i=0}^{6}\pi_i + \sum_{i=7}^{\infty} 0{,}0127635(0{,}3476)^{i-6} + 1{,}49242\,10^{-6}(-0{,}1325)^{i-6} \\ &= \sum_{i=0}^{6}\pi_i + 0{,}0127635\sum_{i=7}^{\infty} 0{,}3476^{i-6} + 1{,}49242\,10^{-6}\sum_{i=7}^{\infty}(-0{,}1325)^{i-6} \\ &= \sum_{i=0}^{6}\pi_i + 0{,}0127635\sum_{j=1}^{\infty} 0{,}3476^{j} + 1{,}49242\,10^{-6}\sum_{j=1}^{\infty}(-0{,}1325)^{j}\end{aligned}$$

La suma de la serie geométrica, al ser su razón menor de 1, converge y puede calcularse con ayuda de la fórmula

$$\sum_{i=n}^{\infty}\rho^i = \frac{\rho^n}{1-\rho} \qquad \rho < 1$$

Por otro lado, la suma de los términos $\pi_0, \ldots, \pi_6$ es conocida, puesto que tenemos los valores numéricos de las primeras componentes de $\boldsymbol{\pi}$. Con todo ello, calculamos finalmente que la suma del vector $\boldsymbol{\pi}$ es:

$$\sum_{i=0}^{\infty}\pi_i = 1{,}353413768$$

Por tanto, dividiendo todos los valores obtenidos para $\pi_0, \pi_1, \ldots, \pi_{12}$ entre esta cantidad, tendremos los valores correctos. Y dividiendo la ecuación (4.34) tendremos la fórmula general para cualquier componente a partir del sexto.

El resultado final es, por tanto el que se muestra en el cuadro 4.8. Observar que los términos $\pi_7, \pi_8, \ldots, \pi_{12}$, se obtienen de dos formas diferentes, pero deben salir lo mismo en ambas. Por un lado salen del sistema de 12 ecuaciones, y por otro lado salen de la fórmula general cerrada válida para $i > 6$.

Conclusiones El método desarrollado en detalle en el ejemplo anterior, es aplicable a cualquier sistema con $\bar{U} < 1$. Sin embargo hay que destacar la complejidad computacional de la solución, a la que contribuyen los siguientes factores:

- Por un lado es necesario computar r columnas de la matriz de Markov, y cada columna exige aplicar el método "convolucionar y encoger" a lo largo de todo un hiperperiodo. Cuantos más trabajos se ejecuten en un hiperperiodo, más costosa será la obtención de cada columna de $\mathbf{P}$.

Primeros 12 componentes					
π_0	0.738872	π_4	0.007869	π_8	0.000114
π_1	0.158917	π_5	0.002705	π_9	0.000040
π_2	0.068203	π_6	0.000944	π_{10}	0.000014
π_3	0.021987	π_7	0.000328	π_{11}	0.000005

Fórmula general para los restantes

$$\pi_i = 0{,}000943062(0{,}34766568)^{i-6} + 1{,}1027\,10^{-6}(-0{,}1324854)^{i-6} \qquad i > 6$$

Cuadro 4.8.: Distribución estacionaria normalizada

- Por otro lado, es necesario diagonalizar la matriz **A**. La dimensión de esta matriz es m_r, que es la longitud de la columna r-ésima de **P**. Esta columna es el resultado de la convolución de todos los tiempos de ejecución de los trabajos a lo largo de un hiperperiodo por lo que, de nuevo, cuantos más trabajos haya mayor será la matriz **A** y el coste de su diagonalización.

- Finalmente es necesario resolver un sistema de $m_r + r$ ecuaciones con $m_r + r$ incógnitas, lo cual de nuevo es tanto más costoso cuanto mayor sea el número de trabajos lanzados en un hiperperiodo.

En el capítulo 6.1 se realizará un análisis detallado de la complejidad computacional del problema, pero ya se puede apuntar que es tan elevada que este método "analítico" para obtener la distribución estacionaria resulta prohibitivo, salvo para sistemas "de juguete". Por ello desarrollaremos (en la sección 4.6) algunos métodos aproximados para resolver el problema más rápidamente.

De todas formas, la solución analítica tiene un cierto interés teórico, y es que nos da la forma general que tiene la distribución estacionaria. Vemos que en todos los casos se compondrá de unos cuantos puntos iniciales, cuya distribución es arbitraria y depende de los parámetros del sistema, y seguidamente viene una cola que tiende a cero. La forma general de esta cola es una suma de exponenciales, y las razones de estas exponenciales son las raíces del polinomio (4.32) que tienen módulo menor de 1. Observar que es posible que algunas de estas razones sean complejas, en cuyo caso la cola tendrá una forma "ondulada" debido a las componentes senoidales, que se irá amortiguando.

4.6. Métodos aproximados para la obtención de la distribución estacionaria

4.6.1. Problemas de la solución analítica

La solución analítica presentada en la página 105 tiene dos graves problemas que reducen su aplicabilidad en la práctica. De un lado, el coste computacional es muy elevado, como ya se apuntó, restringiendo su uso a sistemas en los que el número de trabajos por hiperperiodo no sea demasiado grande, y en el que la PF de los trabajos esté especificada por un número pequeño de puntos. El coste computacional se analizará teóricamente con más detalle en la sección 6.1.

Pero hay otro problema añadido que puede afectar incluso a los sistemas pequeños haciendo imposible su resolución por computador. El método descrito requiere resolver un sistema de ecuaciones, que en la mayor parte de los casos estará muy mal condicionado (es decir, una ligera variación en uno cualquiera de sus coeficientes puede llevar a una solución muy diferente).

Esto se debe a que el sistema a resolver contiene ecuaciones con coeficientes extremadamente pequeños (del orden de 10^{-10}, por ejemplo) junto con otros extremadamente grandes (del orden de 10^{10}, por ejemplo). Al resolver este tipo de sistemas en un computador que, típicamente, utiliza un formato de coma flotante para almacenar los coeficientes, aparecen errores de redondeo numérico. Un formato de coma flotante no puede ser a la vez muy preciso y almacenar números muy grandes. Estos errores numéricos pueden causar que la solución obtenida a través del computador sea muy diferente de la solución "verdadera".

El origen del problema se puede encontrar en cómo se construye el sistema de ecuaciones a resolver. El lector recordará que las $(m_r + 1)$ primeras ecuaciones se obtenían de la matriz $\mathbf{P}$, por lo que los coeficientes de estas ecuaciones serán menores que 1, y típicamente muy pequeños, pues son el resultado de convoluciones. Las restantes r ecuaciones se obtenían de la relación de recurrencia que se originaba en la columna r de la matriz. Esta recurrencia usa los coeficientes de la columna r-ésima, y al despejar de la relación el término con subíndice más alto, es necesario dividir por el coeficiente que afecta a este término. Este coeficiente es el que llamábamos $b_{m_r}(r)$, véase eq. (4.16), y representa la probabilidad de que en un hiperperiodo la carga total sea la máxima posible. Es por tanto el producto de las probabilidades del peor caso para todos los trabajos del hiperperiodo. Se comprende que su valor ha de ser extremadamente pequeño en casos prácticos. Al despejar este término, y dividir por un coeficiente tan próximo a cero, se obtienen coeficientes con valores numéricos muy grandes, véase eq. (4.27), que aparecerán en la última fila de la matriz $\mathbf{A}$,

Esta combinación de coeficientes muy pequeños y muy grandes, hace que el sistema sea casi imposible de resolver numéricamente por computador.

Por estas razones se hace necesario investigar en métodos que permitan obtener una aproximación de la solución buscada, que no presenten estos inconvenientes, es decir, métodos con menor complejidad computacional y menor sensibilidad a los errores numéricos.

En esta sección se presentan dos aproximaciones bastante triviales, pero este podría ser un campo abierto a una investigación más exhaustiva.

4.6.2. Método de truncación de la matriz de Markov

Recordemos que el problema a resolver era despejar $\boldsymbol{\pi}$ en la ecuación $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$, y que esto no era inmediato debido a que la matriz $\mathbf{P}$ era infinita. Una aproximación obvia consistirá en truncar la matriz $\mathbf{P}$, dejándola en una matriz finita $\mathbf{P}'$ de dimensión $M \times M$, eliminando simplemente todo lo que esté más allá de la fila $M-1$ y la columna $M-1$.

De este modo planteamos la nueva ecuación aproximada $\pi' = \mathbf{P}'\boldsymbol{\pi}'$. Resolver esta ecuación se convierte en un problema de hallar el vector propio de $\mathbf{P}'$ correspondiente al valor propio 1. Este problema se puede resolver numéricamente aplicando cualquiera de los métodos en la literatura de algoritmos numéricos (por ejemplo, los contenidos en [Press *et al.*, 1992]).

Ahora bien, puesto que $\mathbf{P}' \neq \mathbf{P}$, es muy posible que $\mathbf{P}'$ no tenga ningún valor propio igual a 1. En este caso elegiremos el más próximo a 1, y nos quedaremos con el vector propio correspondiente a ese valor.

Es evidente que $\boldsymbol{\pi}'$ no es más que una aproximación de $\boldsymbol{\pi}$, en primer lugar porque $\boldsymbol{\pi}'$ sólo tiene M elementos, mientras que $\boldsymbol{\pi}$ tenía infinitos, y en segundo lugar porque $\boldsymbol{\pi}'$ no corresponde a un valor propio de 1, sino a una aproximación. También resulta evidente que, cuanto mayor sea M, mejor será la aproximación.

Esta aproximación reduce la complejidad del problema, aunque no demasiado, ya que sigue siendo necesario de todas formas obtener la matriz $\mathbf{P}'$, lo cual requiere aplicar el método "convolucionar y encoger" M veces a lo largo de todo un hiperperiodo, y resolver un sistema de M ecuaciones con M incógnitas para despejar π'. Su principal ventaja sobre el método "analítico" es que no requiere la matriz $\mathbf{A}$, ni su diagonalización, con el consiguiente ahorro de tiempo, y sobre todo que es mucho más robusto frente a errores numéricos en los coeficientes de $\mathbf{P}'$.

Sin embargo plantea un grave problema, y es que el vector $\boldsymbol{\pi}'$ hallado por truncación, no coincide en ningún punto con la solución $\boldsymbol{\pi}$ "auténtica". Es decir, al truncar $\mathbf{P}$, no obtenemos una "versión truncada" de $\boldsymbol{\pi}$, sino una solución diferente. El error introducido al truncar $\mathbf{P}$, se manifiesta en todos los puntos de $\boldsymbol{\pi}'$. El principal problema es que no se puede predecir de qué forma influirá sobre el análisis posterior este error introducido. Es decir, como consecuencia de esta truncación, el análisis posterior dará unas probabilidades de perder plazos dife-

rentes a las "reales", pero no se puede predecir si las aproximaciones estarán por encima o por debajo.

Por tanto, esta técnica sirve tan sólo para obtener un valor aproximado de la probabilidad de perder plazos, que puede usarse como métrica de calidad para un sistema de tiempo real blando. En cambio no puede usarse para obtener una cota superior a esta probabilidad, que sería útil para sistemas más estrictos.

4.6.3. Método iterativo

Otro enfoque diferente, que ni siquiera requiere obtener la matriz $\mathbf{P}$, sería simplemente aplicar el método "convolucionar y encoger" a lo largo de N hiperperiodos. Al final de cada hiperperiodo se obtiene una distribución de la carga, que se usa como carga inicial para el hiperperiodo siguiente. Hemos demostrado que si $\bar{U} < 1$ esta carga irá convergiendo hacia $\boldsymbol{\pi}$. Por tanto basta comparar la PF que hemos obtenido para el hiperperiodo j con la que habíamos obtenido para el $(j-1)$. Cuando sean lo suficientemente parecidas (esto es, cuando su diferencia cuadrática sea menor que un cierto ϵ preestablecido), dejaremos de iterar, y tomaremos la última PF obtenida como una aproximación de $\boldsymbol{\pi}$.

Este método tiene dos ventajas importantes. En primer lugar, no requiere obtener la matriz de Markov, por lo que su coste computacional, en general, es muchísimo menor que la solución exacta, e incluso que el método de truncación. Además, es muy robusto frente a los errores de redondeo numérico.

Sus inconvenientes son también dos. Por un lado no conocemos de antemano cuántas veces será necesario iterar hasta lograr la convergencia. Cabe esperar que cuanto más próximo a 1 sea el valor de $\bar{U}$, mayor será el número de iteraciones necesarias, y los experimentos así lo han confirmado. Si $\bar{U}$ es casi 1 (del orden de 0.995 por ejemplo), el número de iteraciones que requiere este método puede ser muy grande, tanto que podría resultar más efectivo calcular la matriz de Markov. Sin embargo, para valores de $\bar{U}$ por debajo de 0.9 el método converge sorprendentemente rápido.

El segundo inconveniente es similar al que se planteó para el método de truncación. Y es que la aproximación obtenida no sirve para obtener cotas superiores de la probabilidad de perder plazos. De hecho, en este caso, la probabilidad de perder un plazo que se obtendría a través de esta aproximación es siempre *menor* que la "verdadera" probabilidad. Esto se debe a lo siguiente. Imaginemos que el método converge tras N iteraciones, y da lugar a una aproximación $\boldsymbol{\pi}'$. Esta aproximación tendrá un número finito de elementos, puesto que es el resultado de un número finito de convoluciones. Digamos que tiene n elementos. De acuerdo con esta distribución, la probabilidad de que la carga del sistema tome un valor mayor de $n-1$ sería cero. Sin embargo, la solución "real" tiene infinitos términos, por lo que, siguiendo la verdadera distribución, la probabilidad de que la carga

tome un valor mayor de $n-1$ es no nula. Esto implica que los tiempos de respuesta "reales" pueden tomar valores más altos que los obtenidos a partir de la aproximación, puesto que parten de un sistema con condiciones iniciales peores.

El método iterativo, por tanto, es útil para obtener una cota inferior a la probabilidad de perder plazos. Esta cota es muy ajustada, prácticamente igual a la "verdadera" probabilidad de perder plazos, pero no obstante, desde un punto de vista teórico, está por debajo, por lo que no podría usarse para tomar decisiones de planificabilidad en un sistema de tiempo real estricto.

A modo de ejemplo, el cuadro 4.9 muestra el resultado de aplicar el método iterativo sobre el sistema de ejemplo del cuadro 4.7, que ya fue resuelto por el método "exacto". La última columna de la tabla reproduce de nuevo la solución obtenida por el método exacto, con fines de comparación. La figura 4.21 muestra cómo evoluciona el error cuadrático entre cada iteración y la siguiente, en escala logarítmica. Vemos que la gráfica se asemeja a una recta, lo que apunta a un decrecimiento exponencial en el error cuadrático.

	$\mathcal{W}_0$	$\mathcal{W}_1$	$\mathcal{W}_2$	$\mathcal{W}_3$	$\mathcal{W}_5$	$\mathcal{W}_{10}$	$\mathcal{W}_{20}$	$\mathcal{W}_\infty$
0	1	0.837500	0.789734	0.768523	0.750897	0.740816	0.738968	0.738872
1	-	0.131250	0.150109	0.155394	0.158160	0.158899	0.158919	0.158917
2	-	0.031250	0.050976	0.059129	0.065050	0.067794	0.068186	0.068203
3	-	-	0.008203	0.013632	0.018639	0.021485	0.021964	0.021987
4	-	-	0.000977	0.002906	0.005524	0.007464	0.007850	0.007869
5	-	-	-	0.000385	0.001372	0.002430	0.002690	0.002705
6	-	-	-	0.000030	0.000299	0.000779	0.000934	0.000944
7	-	-	-	-	0.000053	0.000238	0.000321	0.000328
8	-	-	-	-	0.000007	0.000069	0.000110	0.000114
9	-	-	-	-	0.000000	0.000019	0.000037	0.000040
10	-	-	-	-	0.000000	0.000005	0.000013	0.000014
11	-	-	-	-	-	0.000000	0.000004	0.000005
12	-	-	-	-	-	0.000000	0.000001	0.000001
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Cuadro 4.9.: Evolución de la función de probabilidad de la carga por el método iterativo

4.7. Análisis de los trabajos no-basales. Retroceder y Rehacer

Las ideas de "trabajo basal" y "trabajo base" (que se definirá en esta sección), son clave para poder aplicar el análisis antes descrito a los sistemas planificados

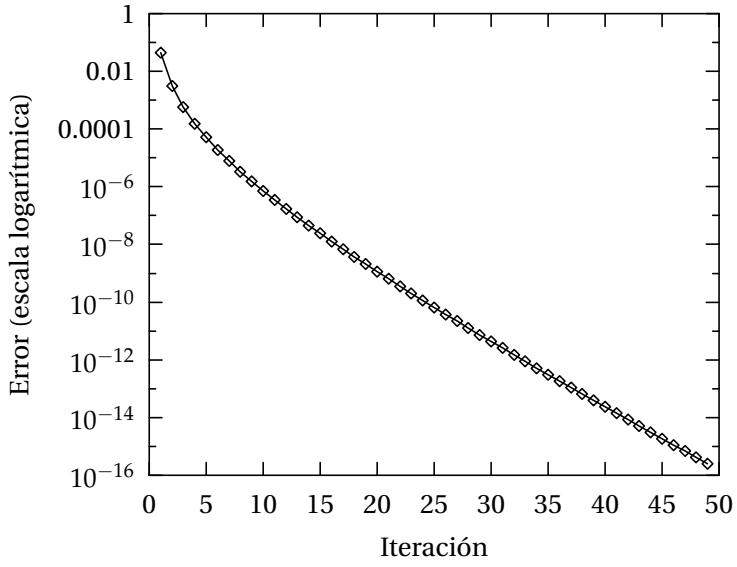


Figura 4.21.: Decrecimiento del error con cada iteración

mediante EDF. Estas ideas son originales de Kanghee Kim (Universidad de Korea) y Lucia Lo Bello (Universidad de Catania). Estos autores y yo hemos establecido una relación de colaboración como consecuencia de la similaridad de algunos de nuestros enfoques, y hemos publicado conjuntamente un artículo [Díaz *et al.*, 2002*a*].

Originalmente, mi análisis estaba más centrado en el planificador RM y DM (la aplicación a este caso particular se explica en la página 129), pero gracias a las ideas de Kim y Lo Bello es fácilmente extensible a EDF. Por ello, aunque las ideas no sean originales mías, he considerado de interés incluir este punto en la tesis. Por otro lado, la demostración de la existencia de un trabajo basal bajo EDF (teorema 8) sí es original mía.

4.7.1. Conceptos previos

En las secciones anteriores hemos resuelto el problema para los trabajos basales. Usando las técnicas expuestas en la sección precedente, obtendremos la distribución de la carga estacionaria. Usando esta carga como inicial, podemos obtener la carga en cualquier otro instante del hiperperiodo, en particular en el instante en que un trabajo basal llega. Esta carga será la que hay que tener en cuenta como carga inicial en el algoritmo "partir, convolucionar y juntar", explicado en la sección 4.4 y así obtener la distribución del tiempo de respuesta.

Sin embargo, si el trabajo es no-basal, la carga calculada incluye contribuciones de trabajos de menor prioridad que el considerado, por lo que no toda esta carga se convertirá en retraso. Es necesario recalcular la distribución que tiene la carga que "realmente" afectará al trabajo en curso, descontando el efecto de los trabajos con menor prioridad que él.

Para ello, lo que haremos será "retroceder" hasta el anterior trabajo basal con prioridad mayor que el considerado. Partiendo de la carga en ese instante, avanzaremos de nuevo usando el algoritmo "convolucionar y encoger", pero esta vez en lugar de ir aplicándolo para cada nuevo trabajo que llega, lo haremos sólo para aquellos de mayor o igual prioridad que el considerado, hasta llegar de nuevo al instante considerado.

Formalicemos estas ideas.

Definición 12. *Llamaremos **trabajo base** del trabajo Γ_j al trabajo basal anterior a λ_j que tenga prioridad mayor o igual que P_j, y con instante de activación más próximo a λ_j.*

Aclaremos el concepto con un ejemplo. Supongamos que tenemos una secuencia de 8 trabajos, $\Gamma_1, \ldots, \Gamma_8$, cada uno de ellos con su prioridad P_i, que se muestra en el cuadro 4.10. No nos importa cómo han sido asignadas estas prioridades, ni tampoco los instantes precisos de activación de cada trabajo, con tal de que estos instantes estén ordenados, es decir, que $\lambda_i \leq \lambda_{i+1}$.

Γ_i	P_i
Γ_1	10
Γ_2	5
Γ_3	8
Γ_4	7

Γ_i	P_i
Γ_5	4
Γ_6	1
Γ_7	2
Γ_8	1

Cuadro 4.10.: Serie de trabajos con diferentes prioridades

La figura 4.22 muestra para cada índice i, la prioridad p_i en una escala vertical, para poder comparar más fácilmente las prioridades entre sí.

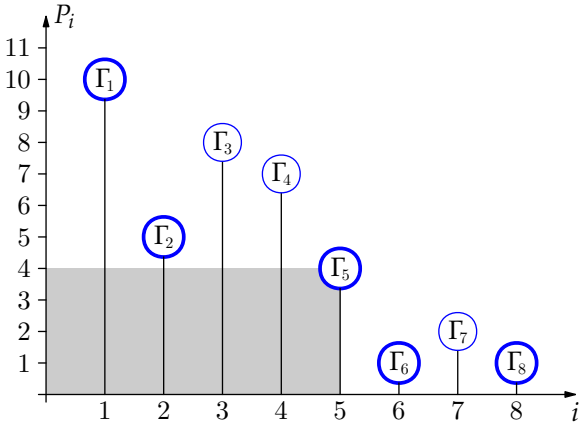


Figura 4.22.: Representación gráfica de las prioridades

Los trabajos basales, tal como se han definido en 2, son los que tienen menor o igual prioridad que todos los que habían llegado antes. Si en la figura 4.22 dibujáramos un rectángulo infinito hacia la izquierda, que tenga como esquina superior

derecha el punto que representa al trabajo, este rectángulo no contendría ningún otro trabajo. En la figura se muestra esta situación para el trabajo Γ_5, que por tanto sería un trabajo basal. El lector puede comprobar que los trabajos $\Gamma_1, \Gamma_2, \Gamma_5, \Gamma_6$ y Γ_8 son basales (el primer trabajo de la secuencia siempre es basal, puesto que no tiene otros a su izquierda), mientras que los restantes Γ_3, Γ_4 y Γ_7 son no-basales: Γ_3 deja a su izquierda Γ_2, de menor prioridad, lo mismo para Γ_4, y Γ_7 deja a su izquierda a Γ_6. La figura marca con un círculo más grueso los trabajos basales.

Aplicando la definición 12 al ejemplo de la figura 4.22 vemos que el trabajo base de Γ_3 sería Γ_1, y lo mismo para Γ_4. El trabajo base de Γ_7 sería Γ_5.

4.7.2. Cálculo del retraso experimentado por cada trabajo

Para los trabajos basales, la carga total de sistema existente en el instante de su activación coincide con el retraso que experimentarán. Esta carga se obtiene partiendo del instante inicial y aplicando el método "convolucionar y encoger" a todos los trabajos que vamos encontrando. Este método por tanto se puede aplicar directamente para obtener la carga de Γ_1, Γ_2, Γ_5, Γ_6 y Γ_8.

Para los no-basales, en cambio, el método no vale ya que incluye carga debida a trabajos de menor prioridad. En este caso lo que debe hacerse es "retroceder" hasta su trabajo base, y aplicar el método "convolucionar y encoger" de ese punto en adelante, pero esta vez saltándose todo trabajo de prioridad menor al considerado. Por ejemplo, para el trabajo Γ_3, se buscará su trabajo base, que es Γ_1; se retrocede hasta el instante λ_1, y desde ese punto se aplicará el algoritmo "convolucionar y encoger" hasta llegar de nuevo al instante λ_3 pero esta vez saltándose todo trabajo con prioridad menor de P_3, es decir, saltándose Γ_2. De forma análoga, para calcular el retraso del no-basal Γ_4, se retrocede hasta λ_1, y partiendo de la carga en ese punto, se avanza de nuevo hasta λ_4, saltándose los trabajos con prioridad menor de P_4, es decir, saltándose Γ_2. Finalmente, para el tercer y último trabajo no-basal, Γ_7, se retrocede al instante λ_5 y, partiendo de la carga en ese instante, se avanza de nuevo hasta λ_7, saltándose en este caso el trabajo Γ_6 que prioridad menor de P_7.

La figura 4.23 muestra un "grafo de cómputo" de los retrasos de los diferentes trabajos. Se parte del nodo marcado como un disco negro, en el cual se tiene como dato de partida la PF de la carga en el instante $\lambda_0 = 0$. Una flecha que une dos nodos $i \rightarrow j$ representa una operación "encoger" de magnitud $(\lambda_j - \lambda_i)$. A la izquierda de un nodo i, se tiene la PF de la carga "justo antes" del instante λ_i, y a su derecha "justo después". Por tanto cada nodo i representa una convolución entre la PF que tiene a su izquierda y la función $f_{\mathcal{C}_i}$.

La parte superior del grafo es la que va calculando la carga total de sistema, aplicando "convolucionar y encoger" sobre todos los trabajos que llegan. Sobre esta rama del grafo estarán los trabajos basales, marcados con un borde más grue-

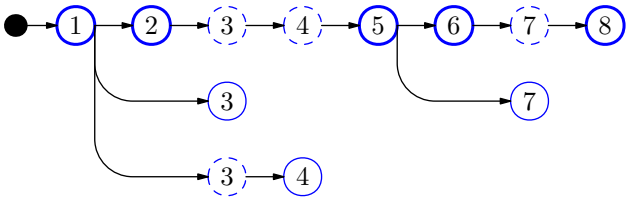


Figura 4.23.: Grafo de cómputo de la carga-retraso

so. Los círculos con líneas discontinuas representan operaciones intermedias que son necesarias, pero que no representan el retraso de ningún trabajo en particular. El retraso experimentado para los trabajos no-basales se obtiene siguiendo otra rama del grafo, que parte del trabajo base correspondiente. Los círculos de línea continua fina representan los trabajos no-basales. Así, por ejemplo, para obtener el retraso que sufrirá el trabajo Γ_7, siguiendo el grafo anterior, vemos que será necesario aplicar el método "convolucionar y encoger" desde el instante inicial hasta el instante λ_5, y finalmente convolucionar con $f_{\mathcal{C}_5}$ y "encoger" $(\lambda_7 - \lambda_5)$ unidades.

Este grafo muestra algo interesante, y es que, en este ejemplo, podemos ahorrarnos una de las convoluciones. En efecto, la rama que indica cómo calcular el retardo de Γ_4 (rama inferior de la figura) tiene gran parte en común con la que calcula el retardo de Γ_3. Esto se debe a que, entre Γ_3 y Γ_4 no llegan trabajos de menor prioridad que Γ_4, como se ve en la figura 4.22. Por tanto podríamos "optimizar" el grafo anterior si aprovechamos el resultado que se obtuvo para Γ_3 a la hora de computar Γ_4. El nuevo grafo "optimizado" sería el mostrado en la figura 4.24.

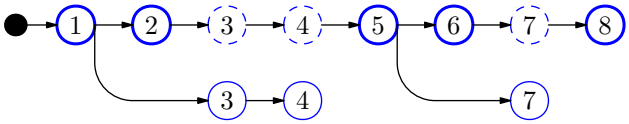


Figura 4.24.: Grafo de cómputo de la carga-retraso "optimizado"

Por tanto, el método general para hallar el tiempo de respuesta de todos los trabajos del sistema sería:

1. Se clasifican todos los trabajos del hiperperiodo en basales y no-basales
2. Se busca el primer trabajo basal que aparece en el hiperperiodo. Si el primer trabajo basal no coincide con el primer trabajo del hiperperiodo, se desplaza el origen del hiperperiodo para que lo sea.
3. Se aplica el análisis Markoviano para determinar la carga de régimen permanente existente en el instante de activación de este trabajo basal.
4. Para cada trabajo basal, se obtiene la carga existente en el instante de su activación, por el método "convolucionar y encoger", tomando como dato de partida la carga existente en el trabajo basal anterior. Una vez obtenida la carga en el instante de activación del trabajo, se aplica la técnica "partir,

convolucionar y juntar" para obtener el tiempo de respuesta de ese trabajo basal.

5. Para cada trabajo no-basal, se obtiene el retraso que sufrirá por el método "convolucionar y encoger", pero en este caso se parte de la carga existente cuando llegó su trabajo base, y las convoluciones se aplican sólo a los trabajos de mayor prioridad que el que se está considerando. Una vez obtenido este retraso inicial, se aplica la técnica "partir, convolucionar y juntar" para obtener el tiempo de respuesta de este trabajo no-basal.

En el procedimiento anterior quedan dos importantes cuestiones en el aire. En primer lugar ¿es posible encontrar siempre un trabajo basal en cualquier hiperperiodo que nos sirva como "punto de partida" a partir del cual derivar el retraso para cualquier otro trabajo? Y en segundo lugar, para los trabajos no-basales ¿es posible encontrar siempre un trabajo base anterior a él que nos permita recalcular el retraso que sufrirá? Es decir, queda en el aire la cuestión de la existencia de los trabajos basales y de los trabajos base. Esta cuestión se clarifica a continuación.

4.7.3. Existencia de los trabajos basales y los trabajos base

Teorema 6. *Dado un sistema S para el cual la secuencia relativa de prioridades entre sus trabajos se repite periódicamente, con periodo T, es siempre posible encontrar al menos un trabajo basal en el primer hiperperiodo. Además, los trabajos basales encontrados en el primer hiperperiodo volverán a ser basales en todos los hiperperiodos siguientes.*

Demostración. Supongamos que todos los trabajos en un hiperperiodo tienen diferente prioridad. Si no fuera el caso, siempre se pueden re-asignar las prioridades de modo que todos las tengan diferentes y se respete la política de que el primero en llegar es más prioritario.

Si todas las prioridades son diferentes, siempre podremos encontrar en un hiperperiodo cualquiera, un trabajo Γ_j que tenga la menor prioridad, P_j, de todas. Este trabajo, que llega en el instante λ_j, tiene menor prioridad que todos los trabajos anteriores a él en ese hiperperiodo, es decir, que todos los trabajos que llegaron en el intervalo $[kT, \lambda_j)$. Es más, tiene también menor prioridad que los que habían llegado antes, en el intervalo $[\lambda_j - T, kT)$. Esto se debe a que, al tener Γ_j la prioridad mínima dentro de su hiperperiodo, su anterior instancia[6] en el instante $\lambda_j - T$ también tendrá la prioridad mínima en el hiperperiodo anterior (puesto que por hipótesis se mantienen la prioridades relativas). Por tanto todos los trabajos que lleguen en $[\lambda_j - T, kT)$ tienen prioridad mayor que la anterior instancia

[6] No nos referimos aquí a instancias de la tarea a la que pertenece el trabajo, sino por así decir al "mismo" trabajo en un hiperperiodo anterior

de Γ_j. Pero esta instancia a su vez también tiene prioridad mayor que Γ_j, debido a la política de "primero en llegar, primero en servirse", de donde se deduce que todos los trabajos llegados entre $[\lambda_j - T, \lambda_j)$ tienen mayor prioridad que Γ_j.

Repitiendo este razonamiento hacia atrás, llegamos a la conclusión de que Γ_j tiene menor prioridad que todos los trabajos que llegaron antes que él, y por tanto es un trabajo basal. De modo que siempre es posible encontrar uno.

Por otro lado, el mismo razonamiento utilizado para probar que es un trabajo basal, prueba que también son basales sus anteriores instancias en hiperperiodos previos. Y a la inversa, si un trabajo es basal en el primer hiperperiodo, también lo será en todos los siguientes. □

Por tanto, para encontrar los trabajos basales en un hiperperiodo basta con fijar un intervalo de tiempo cualquiera $[t, t+T)$ y buscar el trabajo Γ_j de prioridad mínima en ese intervalo. Ese será un trabajo basal. Posteriormente, se analiza una ventana de tamaño T que comience en el instante de activación λ_j de dicho trabajo, y se compara la prioridad de cada nuevo trabajo que llega en esa ventana con la de los que han llegado antes en esa misma ventana. Si es menor que todos, también será basal, sino no. Es decir, que para clasificar los trabajos en basales y no-basales no es necesario mirar los que llegan antes de λ_j.

Abordemos a continuación el problema de la existencia o no existencia de los trabajos base.

Teorema 7. *Dado un sistema S para el cual la secuencia relativa de prioridades entre sus trabajos se repite periódicamente, con periodo T, si fuera posible encontrar un trabajo basal Γ_j tal que su prioridad P_j es mayor que la de todos los trabajos que le siguen, entonces siempre se podrá encontrar el trabajo base de cualquier trabajo no-basal que llegue en el intervalo $[\lambda_j, \lambda_j + T)$.*

Demostración. Es sencillo demostrarlo por reducción al absurdo. Supongamos que un trabajo Γ_k no-basal llega en $[\lambda_j, \lambda_j + T)$, y supongamos que no tiene trabajo base en ese intervalo. Esto significaría que en dicho intervalo no existe ningún trabajo basal con prioridad mayor que p_k (ya que esta es la definición de trabajo base). Pero esto contradice la hipótesis, ya que Γ_j cumple estas condiciones. Luego Γ_k tendrá un trabajo base que en el peor de los casos será Γ_j. □

Por tanto, si tenemos un sistema S que cumple el enunciado del teorema 7, será posible para él determinar los tiempos de respuesta de todos sus trabajos, analizando simplemente una ventana del tamaño de un hiperperiodo.

Si no cumple la condición anterior, será necesario buscar el trabajo base en el hiperperiodo previo, y eventualmente puede ser necesario retroceder un cierto número de hiperperiodos para encontrar el correspondiente trabajo base.

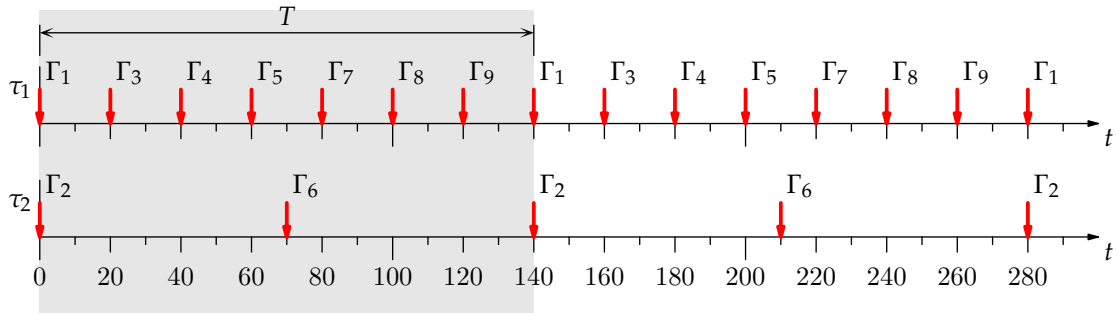


Figura 4.25.: Ejemplo de dos tareas planificadas con EDF

4.7.4. Aplicación a políticas habituales de asignación de prioridades

Observar que para aplicar nuestro análisis se requiere tan sólo que la política de asignación de prioridades asigne éstas de forma predecible y que las prioridades relativas entre los trabajos se repitan periódicamente. Estas condiciones son cumplidas por las bien conocidas políticas RM (*Rate Monotonic*), DM (*Deadline Monotonic*) y EDF (*Earliest Deadline First*). No obstante hay una gran diferencia a la hora de aplicar el análisis a una política que asigna prioridades fijas a las tareas (como RM o DM) y otra que asigna una prioridad diferente a cada instancia de la tarea (como EDF). Analicemos estos casos por separado.

Política EDF

En esta política cada trabajo tiene diferente prioridad aunque pertenezca a la misma tarea, ya que la prioridad se basa en los plazos absolutos de cada trabajo. El trabajo que tenga el plazo más próximo al instante actual será el de mayor prioridad. Esto significa que una tarea que podría ser la de máxima prioridad en un instante dado, puede pasar a ser de menor prioridad cuando llegue otro trabajo con un plazo más cercano.

La figura 4.25 muestra un ejemplo con dos tareas de periodos 20 y 70, y en el que suponemos que el plazo de cada tarea coincide con su periodo. Vemos que los trabajos Γ_1, Γ_3 y Γ_4 pertenecientes a la tarea τ_1 son de mayor prioridad que el trabajo Γ_2, perteneciente a la tarea τ_2, debido a que sus plazos absolutos son menores que el de Γ_2, que ocurre en el instante $t = 70$. Sin embargo, el trabajo Γ_5 tiene su plazo absoluto en el instante $t = 80$, por lo que su prioridad es menor que la de Γ_2. Queda pues ilustrado que no todos los trabajos pertenecientes a una misma tarea tienen la misma prioridad.

Si quisiéramos representar gráficamente las prioridades de cada trabajo, debido a que los primeros en llegar suelen tener plazos absolutos más tempranos, tendrán mayor prioridad. A medida que avanzamos en el tiempo, la prioridad "en

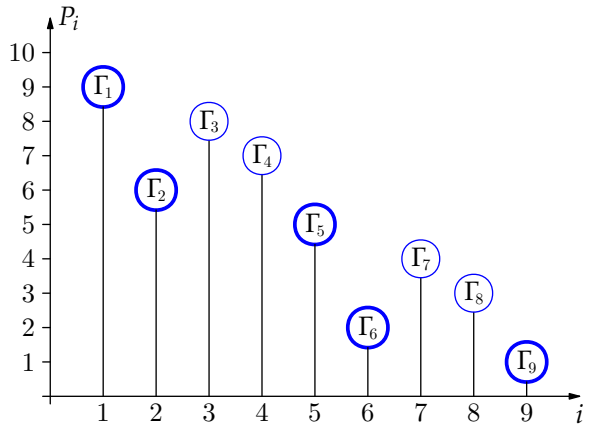


Figura 4.26.: Evolución de las prioridades en el ejemplo anterior

general" de los nuevos trabajos va disminuyendo. Es decir, en cada nuevo hiperperiodo, las prioridades de los trabajos son menores que las de "los mismos" trabajos del hiperperiodo anterior. Aún así se mantiene la prioridad relativa entre los trabajos. Esta situación se ilustra en la figura 4.26.

En EDF hay por tanto una tendencia a que la mayor parte de los trabajos sean trabajos basales, que tienen menor prioridad que todos los que llegaron antes. No obstante aparecen ocasionalmente trabajos no-basales. En el ejemplo anterior, serían trabajos no-basales $\Gamma_3, \Gamma_4, \Gamma_7$ y Γ_8. Los dos primeros tienen su trabajo base en Γ_1 y los dos segundos en Γ_5.

Parece intuitivamente que bajo EDF encontraremos un trabajo base sin necesidad de retroceder mucho, ya que buscamos un trabajo basal (que abundan), que tenga mayor prioridad que el actual (lo que también es probable dada la tendencia de las prioridades a crecer a medida que retrocedemos en el tiempo). Esta intuición es correcta, como queda demostrado en el siguiente teorema.

Teorema 8. *Para un sistema planificado por EDF, siempre es posible encontrar el trabajo base de un trabajo cualquiera dentro de una ventana temporal no mayor que* $D^{max} + T$*, siendo* $D^{max} = \max_i(D_i)$ *y* T *la longitud del hiperperiodo.*

Demostración. Sea τ_i la tarea que tiene el plazo relativo máximo entre todos las del sistema. Sea $\tau_{i,j}$ la instancia de dicha tarea que llega en el instante $\lambda_{i,j}$. El plazo absoluto de este trabajo será $(\lambda_{i,j} + D^{\max})$. Es evidente que todos los trabajos que han llegado antes que éste han de tener un plazo absoluto menor, pues han sido lanzados antes y su plazo relativo no puede ser mayor. Por tanto, bajo EDF, tendrán mayor prioridad. Así pues tenemos un trabajo que tiene menor prioridad que todos los anteriores y por tanto es un trabajo basal.

La siguiente instancia del "mismo" trabajo, llegará en $\lambda_{i,j} + T$, y será también un trabajo basal, puesto que la "basalidad" es periódica por ser periódicas las prioridades relativas de los trabajos.

Sea $\Gamma_{k,l}$ un trabajo no-basal que llega en un instante $\lambda_{k,l}$ tal que $(\lambda_{i,j} + D^{\max}) < \lambda_{k,l} \leq (\lambda_{i,j} + D^{\max} + T)$. Puesto que llega después del instante $(\lambda_{i,j} + D^{\max})$, su plazo absoluto habrá de estar situado también después de ese instante , y por tanto $\Gamma_{i,j}$ tiene mayor prioridad que $\Gamma_{k,l}$. Por tanto, en el peor de los casos, cuando estemos buscando el trabajo base de $\Gamma_{k,l}$ habremos de retroceder como máximo hasta $\Gamma_{i,j}$, puesto que este ya puede servir como trabajo base, al ser un trabajo basal de mayor prioridad que $\Gamma_{k,l}$.

El instante más tardío permitido para $\Gamma_{k,l}$ es $(\lambda_{i,j} + D^{\max} + T)$ y por tanto el tamaño máximo de la ventana a retroceder es $(D^{\max} + T)$. Para todo trabajo que llegue después de $(\lambda_{i,j} + D^{\max} + T)$, podemos usar como trabajo base la instancia de la tarea τ_i que llega en $(\lambda_{i,j} + T)$, usando el mismo razonamiento.

Luego para cualquier trabajo podemos encontrar un trabajo base retrocediendo a lo sumo $(D^{\max} + T)$. □

Como corolario, podemos concluir que en un sistema "razonable" en el que los plazos de las tareas no excedan la duración de un hiperperiodo, siempre encontraremos el trabajo base de cualquier trabajo no-basal dentro de una ventana de tamaño $2T$.

Política RM y DM

En estas políticas ocurre algo diferente. Puesto que todos los trabajos correspondientes a una misma tarea tienen la misma prioridad, esto acarrea las siguientes consecuencias:

1. Todos los trabajos correspondientes a la tarea de menor prioridad son trabajos basales.
2. Todos los restantes trabajos correspondientes a tareas de prioridad distinta de la mínima son no-basales.
3. Es imposible encontrar el trabajo base para cualquiera de los trabajos no-basales.

En efecto, la tercera se deduce de las dos primeras. Todos los trabajos basales tienen menor prioridad que cualquier trabajo no-basal, por tanto será imposible encontrar entre ellos un trabajo base que requiere, por definición, ser de mayor prioridad.

Así pues, bajo RM y DM (y en general bajo cualquier política que asigne prioridades fijas a todos los trabajos de una misma tarea), no es posible construir el grafo de dependencias de la carga entre trabajos basales y no-basales.

La solución a este problema es sencilla si caemos en la cuenta de que, una vez analizados los tiempos de respuesta de los trabajos basales, tendremos calculado el tiempo de respuesta de la tarea de menor prioridad. De modo que podemos

eliminar esta tarea del sistema, dando lugar a un nuevo sistema con una tarea menos. Este nuevo sistema tendrá a su vez una tarea de menor prioridad que podrá ser analizada (puesto que ahora todas sus instancias son trabajos basales al haber eliminado la tarea de menor prioridad). Y así sucesivamente, el sistema puede ser analizado en diferentes fases, considerando en cada una de ellas sólo los trabajos por encima de una prioridad.

Por tanto, el algoritmo para resolver un sistema con planificación de prioridades fijas sería:

1. Para cada tarea del sistema, comenzando por la de menor prioridad
   - *a*) Plantear el sistema formado por todas las tareas de prioridad mayor o igual a la que se considera
   - *b*) En este sistema todos los trabajos de menor prioridad son basales, de modo que se puede:
     1) Obtener la distribución estadística estacionaria de la carga para el primero de ellos por análisis Markoviano
     2) Aplicar el método "convolucionar y encoger" para obtener la carga de los restantes trabajos basales en el hiperperiodo
     3) Aplicar el método "partir, convolucionar y juntar" para obtener la PF del tiempo de respuesta para estos trabajos basales.
   - *c*) Promediando los PF obtenidos, se tiene el PF del tiempo de respuesta para la tarea considerada
2. Eliminar del sistema la tarea ya calculada, avanzar a la siguiente tarea por orden de prioridad creciente y volver al paso 1*a*

Vemos por tanto que la principal diferencia entre la planificación de prioridades fija y la dinámica es que la fija requiere un análisis Markoviano para cada nivel de prioridad del sistema, mientras que la dinámica hace un único análisis Markoviano para la "carga total" y de él puede deducir la carga a otros niveles de prioridad por el mecanismo de los trabajos base.

5. Extensiones al modelo

> If the only tool you have is a hammer, you tend to see every problem as a nail.
>
> *(Abraham H. Maslow)*

En este capítulo plantearemos cómo eliminar ciertas limitaciones del modelo, de forma que el análisis estocástico siga siendo posible.

En todos los casos, un análisis exacto deja de ser viable. En ocasiones porque el modelo del sistema no incorpora suficiente información como para poder predecir la probabilidad exacta de cada tiempo de respuesta, y en otros porque, aunque teóricamente sería posible obtenerla, en la práctica la complejidad computacional involucrada resulta prohibitiva.

De modo que en este modelo ampliado el análisis estocástico deja de ser exacto, esto es, la probabilidad real de que una tarea pierda un plazo es diferente de la que se obtiene en el análisis. Por supuesto interesa que la probabilidad real sea siempre menor que la que se obtiene en el análisis, de forma que este análisis sea útil para tomar decisiones de factibilidad. Es decir, interesa que el análisis garantice ser pesimista. Por esta razón, antes de entrar en las ampliaciones propiamente dichas, comenzaremos por una sección que plantea formalmente qué se entiende por pesimismo en el caso del análisis estocástico, y enuncia algunas propiedades que se utilizarán en el desarrollo posterior.

Las ampliaciones que posteriormente se introducirán permitirán que las tareas interfieran en el acceso a recursos compartidos (bloqueo), la posibilidad de que el instante de llegada de los trabajos esté sometido a una pequeña variación aleatoria (*jitter*), y la posibilidad de que en el sistema existan tareas esporádicas, cuyo instante de llegada es desconocido. También se discute brevemente el problema de la correlación estadística de los tiempos de ejecución de las tareas, que no es contemplado por el modelo.

5.1. Análisis pesimista. Distribuciones “peores”

Supongamos que estamos interesados en conocer la probabilidad de que una cierta tarea del sistema pierda su plazo. Si tuviéramos toda la información necesaria sobre el sistema, y toda la potencia computacional que quisiéramos, podría-

mos hallar la probabilidad exacta de que una tarea del sistema pierda su plazo. Llamemos a esta probabilidad "real" *PPP*.

Pero si nuestro modelo no es tan exacto, o bien el análisis recurre a algún método aproximado para ahorrar tiempo de computación, entonces obtendremos como resultado una probabilidad de perder plazo que puede no coincidir con la "real". Llamemos a la probabilidad de perder plazo obtenida del análisis PPP'.

Es evidente que el analista y el diseñador estarán dispuestos a aceptar PPP' como una aproximación de *PPP* sólo si:

- PPP' viene acompañada de una cota del error cometido
- O bien el análisis garantiza que $PPP' \geq PPP$

El primer caso serviría para sistemas de tiempo real blando, ya que se obtiene una aproximación de la verdadera probabilidad, junto con una información de cómo es de buena la aproximación. En cambio para sistemas de tiempo real más estricto resulta preferible el segundo caso, ya que si construimos un sistema que según el análisis tiene una probabilidad de perder plazos PPP', sabemos que en su funcionamiento real, la frecuencia de pérdida de plazos será aún menor que PPP'.

En este capítulo nos centraremos en el segundo caso. Es decir, las ampliaciones al modelo aquí desarrolladas junto con el análisis propuesto proporcionarán de forma garantizada unas probabilidades de pérdida de plazos que serán mayores que las reales *para todas las tareas del sistema*. El análisis propuesto no proporcionará información sobre cómo de buena es la aproximación. Puede que la probabilidad "real" esté muy lejos de la obtenida del análisis, pero en todo caso garantizaremos que está por debajo, lo cual es el tipo de garantías que necesita el diseñador de un sistema de tiempo real estricto.

Formalizaremos esta idea a través de un operador de comparación de funciones de probabilidad, que nos permita decir si una función de probabilidad es "peor" que otra (en el sentido de proporcionar probabilidades de pérdida de plazo mayores).

5.1.1. Distribuciones "peores"

Definición 13. *Dadas dos variables aleatorias $\mathcal{A}$ y $\mathcal{B}$, diremos que $\mathcal{A}$ es **peor** que $\mathcal{B}$, y lo denotaremos por $\mathcal{A} \succcurlyeq \mathcal{B}$, si entre sus funciones de probabilidad acumuladas se cumple la siguiente relación:*

$$F_{\mathcal{A}}(i) \leq F_{\mathcal{B}}(i) \quad \forall i$$

O lo que es lo mismo:

$$\mathbb{P}\{\mathcal{A} \leq i\} \leq \mathbb{P}\{\mathcal{B} \leq i\} \quad \forall i$$

La figura 5.1 muestra un ejemplo. Para que $\mathcal{A} \succcurlyeq \mathcal{B}$, la función de probabilidad acumulada de $\mathcal{A}$ no debe estar por encima de la de $\mathcal{B}$ en ningún punto (aunque puede coincidir en algunos o en todos).

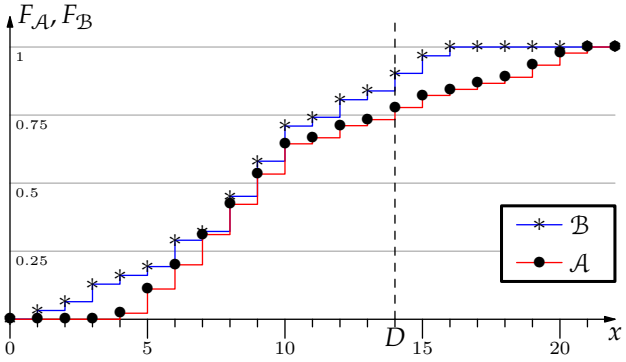


Figura 5.1.: Ejemplo de una distribución "peor" que otra ($\mathcal{A} \succcurlyeq \mathcal{B}$)

Observar que si $\mathcal{A}$ y $\mathcal{B}$ fuesen tiempos de respuesta de diferentes tareas, para cualquier plazo D que fijáramos, se cumpliría que la probabilidad de que $\mathcal{A}$ cumpla el plazo es menor que la de $\mathcal{B}$. O inversamente, $\mathcal{A}$ tiene mayor probabilidad de perder el plazo que $\mathcal{B}$, para cualquier plazo que fijemos. Esto cumple con la condición de pesimismo que queríamos imponer al sistema, y podemos formalizarlo en la siguiente definición.

Definición 14. *Dado un sistema de tiempo real, aplicamos sobre él dos análisis diferentes A y A'. Cada análisis produce como resultado los tiempos de respuesta de todas las tareas del sistema (en realidad sus funciones de probabilidad). Sean $\mathcal{R}_i$ los tiempos de respuesta obtenidos con el análisis A y $\mathcal{R}'_i$ los obtenidos con el análisis A'.*

*Diremos que el análisis A' es más **pesimista** que A si*

$$\mathcal{R}'_i \succcurlyeq \mathcal{R}_i \quad \forall i$$

De la anterior definición de pesimismo y de la definición del comparador "$\succcurlyeq$", se deduce que un análisis más pesimista producirá mayores valores para las probabilidades de pérdida de plazos (*PPP*s), lo que encaja con la definición intuitiva de pesimismo en el análisis.

5.1.2. Propiedades de la relación "peor que"

La relación "peor que" ($\succcurlyeq$) tiene una serie de propiedades que nos serán de utilidad en las secciones siguientes. La demostración de estas propiedades es bastante sencilla en casi todos los casos y por no distraer la atención del lector, se ha relegado a un apéndice.

Por conveniencia, definiremos una variable aleatoria "nula" y llamaremos variables aleatorias positivas a las que son peores que la nula.

Definición 15. *Llamamos variable aleatoria nula, y la denotaremos por* $\mathcal{O}$*, a aquella que sólo puede tomar el valor cero.*

Su función de probabilidad es cero en todos sus puntos, excepto en cero donde toma el valor 1. Su función de probabilidad acumulada presenta un escalón de altura 1 en la ordenada cero.

Definición 16. *Diremos que una variable aleatoria* $\mathcal{A}$ *es positiva, si* $\mathcal{A} \succcurlyeq \mathcal{O}$.

La figura 5.2 muestra un ejemplo de una variable aleatoria positiva, comparando su función de probabilidad acumulada con la de la variable nula $\mathcal{O}$.

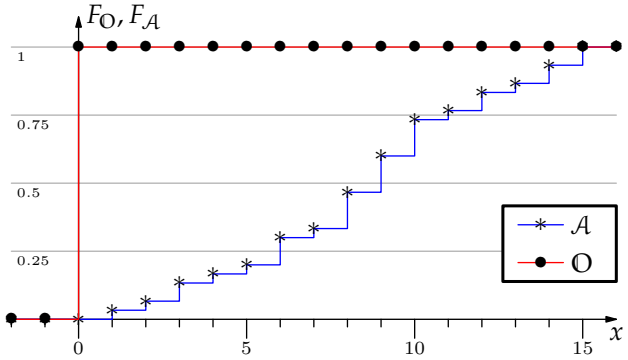


Figura 5.2.: Ejemplo de la distribución de una variable positiva

Esta definición implica que la variable aleatoria $\mathcal{A}$ no puede tomar valores negativos (la probabilidad de que $\mathcal{A}$ sea menor que cero, es cero), lo cual concuerda con la idea intuitiva de variable positiva. Observar que tanto el "tiempo de ejecución", como la "carga pendiente" como el "tiempo de respuesta" son variables positivas.

Algunas propiedades importantes de la relación "peor que" son las siguientes:

P-1 **Reflexiva.** $\mathcal{A} \succcurlyeq \mathcal{A}$.

P-2 **Transitiva.** Si $\mathcal{A} \succcurlyeq \mathcal{B}$ y $\mathcal{B} \succcurlyeq \mathcal{C}$, entonces $\mathcal{A} \succcurlyeq \mathcal{C}$.

P-3 **Orden no completo.** No es cierto que o bien $\mathcal{A} \succcurlyeq \mathcal{B}$ o bien $\mathcal{B} \succcurlyeq \mathcal{A}$. Existen pares de variables aleatorias $\mathcal{A}, \mathcal{B}$ que no son comparables. Por ejemplo, la figura 5.3 muestra un caso así. Ninguna de las dos funciones está por debajo de la otra en todos sus puntos. En general, si las funciones de probabilidad acumulada se cruzan, las variables no son comparables. Si no se cruzan, entonces la variable que corresponde a la gráfica que queda por debajo es peor que la otra.

P-4 **Conservación de la relación "peor que" con la suma.** Si $\mathcal{A} \succcurlyeq \mathcal{B}$, entonces $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{C}$ para toda $\mathcal{C}$ positiva.

Es más, si $\mathcal{A} \succcurlyeq \mathcal{B}$ y también $\mathcal{C} \succcurlyeq \mathcal{D}$, entonces $\mathcal{A} + \mathcal{B} \succcurlyeq \mathcal{C} + \mathcal{D}$

P-5 **Incremento del pesimismo con la suma.** $\mathcal{A} + \mathcal{B} \succcurlyeq \mathcal{A}$ para toda $\mathcal{B}$ positiva.

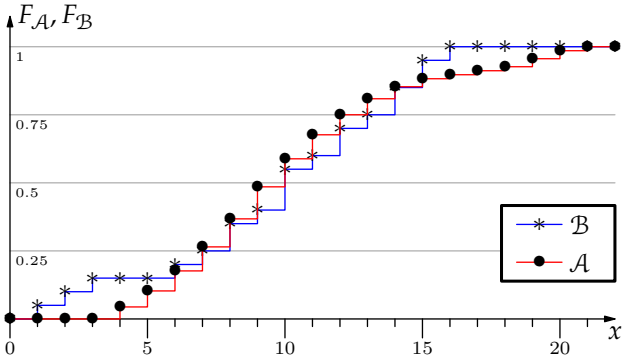


Figura 5.3.: Ejemplo de dos variables aleatorias no comparables

5.1.3. Propiedades del análisis estocástico

Más útiles para nuestros propósitos son las propiedades que cumplen los operadores "desplazar y encoger", y "partir, convolucionar y juntar". Para enunciar formalmente estas propiedades, conviene definir antes estos operadores en forma de función:

Definición 17. *La función Encoge*$(\mathcal{W}, \Delta)$ *convierte una variable aleatoria positiva* $\mathcal{W}$ *en otra variable aleatoria positiva* $\mathcal{W}'$ *:*

$$\mathcal{W}' = \mathit{Encoge}(\mathcal{W}, \Delta)$$

cuya función de probabilidad viene dada por:

$$f_{\mathcal{W}'}(w) = \begin{cases} 0 & \text{para } w < 0 \\ \sum_{i=-\infty}^{\Delta} f_{\mathcal{W}}(i) & \text{para } w = 0 \\ f_{\mathcal{W}}(w + \Delta) & \text{para } w > 0 \end{cases}$$

La definición de este operador implica que la función de probabilidad acumulada de $\mathcal{W}'$ se construye simplemente desplazando a la izquierda Δ unidades la de $\mathcal{W}$, y haciendo cero todo lo que quede a la izquierda del origen (ver fig. 5.4). Por tanto es trivial ver que cumple las propiedades siguientes:

P-6 Conservación de la relación "peor que" al encoger. Si $\mathcal{A} \succcurlyeq \mathcal{B}$, entonces $\text{Encoge}(\mathcal{A}, \Delta) \succcurlyeq \text{Encoge}(\mathcal{B}, \Delta)$, para cualquier valor de $\Delta \geq 0$. Esto es así porque las funciones de probabilidad acumulada de $\mathcal{A}$ y $\mathcal{B}$ se desplazan a la izquierda solidariamente la misma cantidad Δ, manteniendo por tanto las distancias relativas entre todos sus puntos, como se muestra en la figura 5.4.

P-7 Decremento del pesimismo al encoger. La variable aleatoria original $\mathcal{A}$ es siempre peor que su versión encogida. Y entre dos versiones encogidas, es peor la que ha sufrido un encogimiento menor. Es decir, $\mathcal{A} \succcurlyeq \text{Encoge}(\mathcal{A}, \Delta_1) \succcurlyeq \text{Encoge}(\mathcal{A}, \Delta_2)$ para $\Delta_2 \geq \Delta_1 \geq 0$. El por qué se comprende fácilmente observando la figura 5.5

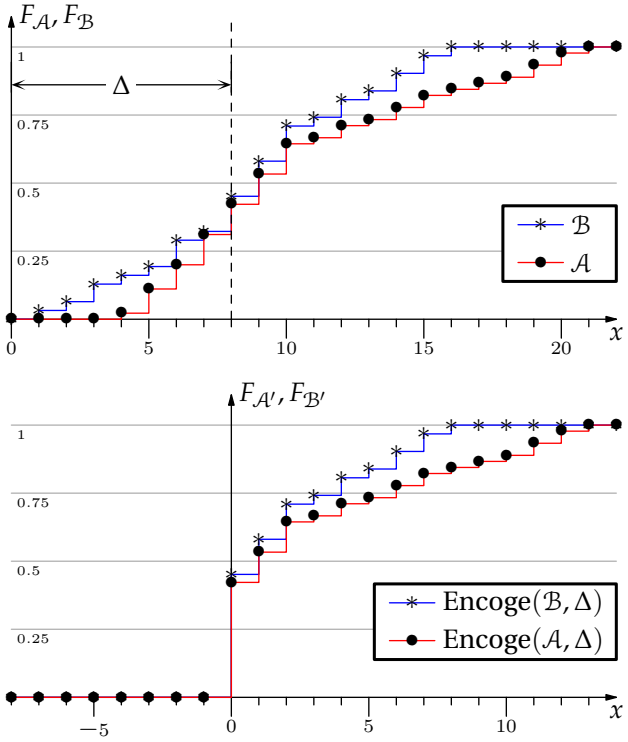


Figura 5.4.: Efecto del operador "encoger" sobre la función de probabilidad acumulada

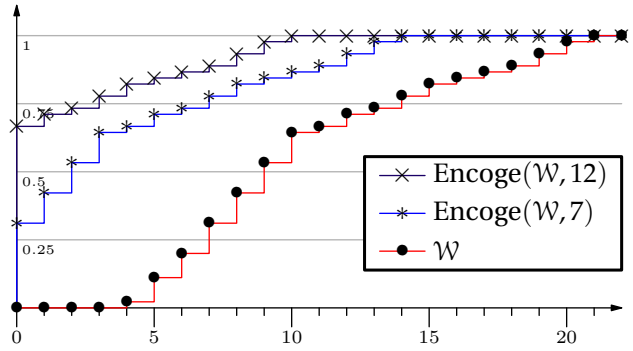


Figura 5.5.: Efecto del operador "encoger" con diferentes valores del parámetro Δ

P-8 Conservación de la relación "peor que" con el método "partir, convolucionar y juntar". Si aplica la misma secuencia de operaciones "partir, convolucionar y juntar" a dos variables aleatorias $\mathcal{R}_j^{\langle 0\rangle}$ y $\mathcal{R}_k^{\langle 0\rangle}$ tales que $\mathcal{R}_k^{\langle 0\rangle} \succcurlyeq \mathcal{R}_j^{\langle 0\rangle}$, en cada paso del proceso se mantiene la relación, esto es: $\mathcal{R}_k^{\langle m\rangle} \succcurlyeq \mathcal{R}_j^{\langle m\rangle}$ para $m = 0, 1, 2, \ldots$

P-9 Incremento del pesimismo con el método "partir, convolucionar y juntar". En cada iteración de este algoritmo, el tiempo de respuesta que se obtiene es "peor" que el de la iteración anterior. Es decir:

$$\mathcal{R}_j^{\langle k\rangle} \succcurlyeq \mathcal{R}_j^{\langle k-1\rangle}$$

Esto es intuitivamente claro, ya que en cada nueva iteración tomamos en cuenta la interferencia de otro trabajo más, y por tanto los tiempos de respuesta tendrán probabilidades mayores de ser más altos.

En base a las propiedades anteriores podemos probar un par de hechos que intuitivamente son evidentes, pero que es conveniente tener demostrados para poder hacer uso de ellos más adelante. Se enuncian y demuestran en las proposiciones siguientes.

Proposición 4. *Dados dos sistemas idénticos (con los mismos parámetros en todas sus tareas), pero con diferentes condiciones iniciales,* $\mathcal{W}(0)$ *y* $\mathcal{W}'(0)$*, si* $\mathcal{W}'(0) \succcurlyeq \mathcal{W}(0)$*, entonces* $\mathcal{W}'(t) \succcurlyeq \mathcal{W}(t)$ *para cualquier instante posterior* t*.*

Demostración. Supongamos que llega un trabajo en $t = 0$, con un tiempo de computación $\mathcal{C}_0$, y que el siguiente trabajo llega en $t = \lambda_1$. Para obtener la carga en cualquier instante antes de que llegue el siguiente trabajo, $t \leq \lambda_1$, es necesario aplicar el algoritmo "convolucionar y encoger", convolucionando primero con $\mathcal{C}_0$ y encogiendo seguidamente en t unidades. La nueva variable aleatoria $\mathcal{W}(t)$ será entonces:

$$\mathcal{W}(t) = \text{Encoge}(\mathcal{W}(0) + \mathcal{C}_0, t) \quad t \leq \lambda_1$$

En el segundo sistema la carga inicial es $\mathcal{W}'(0)$, pero el resto de los parámetros son idénticos (los instantes de llegada de los trabajos, las funciones de probabilidad de sus tiempos de ejecución), de modo que llegamos a una expresión análoga:

$$\mathcal{W}'(t) = \text{Encoge}(\mathcal{W}'(0) + \mathcal{C}_0, t) \quad t \leq \lambda_1$$

Por hipótesis $\mathcal{W}'(0) \succcurlyeq \mathcal{W}(0)$, por tanto $\mathcal{W}'(0) + \mathcal{C}_0 \succcurlyeq \mathcal{W}(0) + \mathcal{C}_0$, en virtud de la propiedad P-4. Así pues, usando la propiedad P-6, tenemos que $\text{Encoge}(\mathcal{W}'(0) + \mathcal{C}_0, t) \succcurlyeq \text{Encoge}(\mathcal{W}(0) + \mathcal{C}_0, t)$, lo que demuestra $\mathcal{W}'(t) \succcurlyeq \mathcal{W}(t)$ para $t \leq \lambda_1$.

Usando este resultado en $t = \lambda_1$ podemos demostrar de forma análoga que se cumple también para $t \leq \lambda_2$, y así sucesivamente. □

Proposición 5. *Sean dos trabajos Γ_k y Γ_j con el mismo tiempo de computación $\mathcal{C}_k = \mathcal{C}_j$, y la misma secuencia de interferencias (expulsiones), es decir, cumplen $\mathcal{C}_{k+i} = \mathcal{C}_{j+i}$, $(\lambda_{k+i} - \lambda_k) = (\lambda_{j+i} - \lambda_j)$ y $(P_{k+i} - P_k) = (P_{j+i} - P_j)$ para $i = 1, 2, \ldots$ Estos trabajos sólo se diferencian en la carga existente, para su nivel de prioridad, en el instante de su llegada, que denotaremos por $\mathcal{W}_{P_k}(\lambda_k)$ y $\mathcal{W}_{P_j}(\lambda_j)$, respectivamente.*

Si $\mathcal{W}_{P_k}(\lambda_k) \succcurlyeq \mathcal{W}_{P_j}(\lambda_j)$, entonces $\mathcal{R}_k \succcurlyeq \mathcal{R}_j$.

Demostración. La demostración es trivial, ya que para obtener $\mathcal{R}_k$ aplicando el algoritmo "partir, convolucionar y juntar" hay que comenzar por obtener $\mathcal{R}_k^{\langle 0 \rangle} = \mathcal{W}_{P_k}(\lambda_k) + \mathcal{C}_k$, el cual es peor que $\mathcal{R}_j^{\langle 0 \rangle} = \mathcal{W}_{P_j}(\lambda_j) + \mathcal{C}_j$, por ser $\mathcal{C}_k = \mathcal{C}_j$ y $\mathcal{W}_{P_k}(\lambda_k) \succcurlyeq \mathcal{W}_{P_j}(\lambda_j)$ por hipótesis. Por tanto $\mathcal{R}_k^{\langle 0 \rangle} \succcurlyeq \mathcal{R}_j^{\langle 0 \rangle}$, luego, por la propiedad P-8 se cumplirá para toda iteración posterior del algoritmo. Es decir $\mathcal{R}_k^{\langle m \rangle} \succcurlyeq \mathcal{R}_j^{\langle m \rangle}$, y haciendo $m \to \infty$ llegamos a que $\mathcal{R}_k \succcurlyeq \mathcal{R}_j$ □

Estas dos proposiciones, que intuitivamente son muy evidentes, nos dan la justificación teórica para afirmar que el método iterativo visto en 4.6.3 no es un método de análisis que proporcione probabilidades pesimistas, como ya se dijo sin demostrarlo entonces. En efecto, siguiendo la proposición 4 vemos que la carga al comienzo de cada hiperperiodo será "peor" que la que había al comienzo del hiperperiodo anterior. Por tanto, también en cualquier instante intermedio del hiperperiodo, es decir $\mathcal{W}(kT + t) \succcurlyeq \mathcal{W}((k-1)T + t)$. En particular, en los instantes de llegada de cada trabajo la carga que le retrasará será "peor" a medida que avanzamos en los hiperperiodos, por lo que aplicando la proposición 5, su tiempo de respuesta será peor.

De modo que en cada iteración del método iterativo, se obtiene un análisis más pesimista. Por tanto, al detenernos en un hiperperiodo cualquiera, obtenemos un análisis más optimista que el que obtendríamos si siguiéramos iterando, y en particular, que el que obtendríamos aplicando la solución "exacta" para la distribución estacionaria. Por esto el análisis iterativo no es pesimista.

Pero el método iterativo puede ser convertido en pesimista si en lugar de suponer la carga inicial cero, la suponemos igual a un valor $\mathcal{W}_0$, que sea "peor" que la carga de régimen permanente. Es decir, $\mathcal{W}_0 \succcurlyeq \mathcal{W}_\infty$. Aplicando el método iterativo partiendo de esta carga inicial, obtendremos aproximaciones cada vez más cercanas a $\mathcal{W}_\infty$, pero todas ellas serán "peores", por lo que garantizarán tiempos de respuesta pesimistas. El único problema estriba en encontrar un $\mathcal{W}_0$ inicial que sea peor que el estacionario. Este es un tema abierto a la investigación.

5.1.4. Máximo de un conjunto de variables aleatorias

En ocasiones tendremos un conjunto de variables aleatorias $\{\mathcal{X}_i\}$ y querríamos obtener la peor de todas ellas, que denotaríamos por $\text{máx}_i\{\mathcal{X}_i\}$. Sin embargo es

posible que en el conjunto no exista ninguna variable $\mathcal{X}_k$ tal que $\mathcal{X}_k \succcurlyeq \mathcal{X}_i$ para todo i. En este caso podemos crear una variable aleatoria nueva que sí lo cumpla, haciendo que su distribución de probabilidad sea la frontera inferior de las distribuciones de las $\mathcal{X}_i$. Por analogía con el concepto de "supremo" de un conjunto, denotaremos esta variable como $\sup_i\{\mathcal{X}_i\}$, y lo definiremos formalmente como sigue:

Definición 18. *Dado un conjunto de variables aleatorias $\{\mathcal{X}_i\}$, llamaremos "supremo" de ese conjunto y lo denotaremos por $\sup_i\{\mathcal{X}_i\}$ a la variable aleatoria cuya función de distribución se construye en la forma siguiente:*

$$F_{\sup_i\{\mathcal{X}_i\}}(x) = \min_i F_{\mathcal{X}_i}(x)$$

Esta variable aleatoria, por construcción cumple que:

$$\sup_i\{\mathcal{X}_i\} \succcurlyeq \mathcal{X}_i \quad \forall i$$

Por tanto se trata de una variable aleatoria que es peor que cualquiera de las variables $\mathcal{X}_i$. En un abuso del lenguaje nos referiremos en ocasiones a ella como "la peor de todas", si bien debe quedar claro que en realidad esta "peor de todas" puede no pertenecer de hecho al conjunto.

La figura 5.6 muestra un par de ejemplos. En el primero de ellos, la "peor de todas" resulta ser una de las variables del conjunto, pero en el segundo es una variable creada artificialmente que no pertenece al conjunto. En cualquier caso, la definición anterior nos da la forma de encontrar su función de probabilidad acumulada.

El supremo del conjunto es peor también que cualquier ponderación de las variables del conjunto. Definamos esto formalmente:

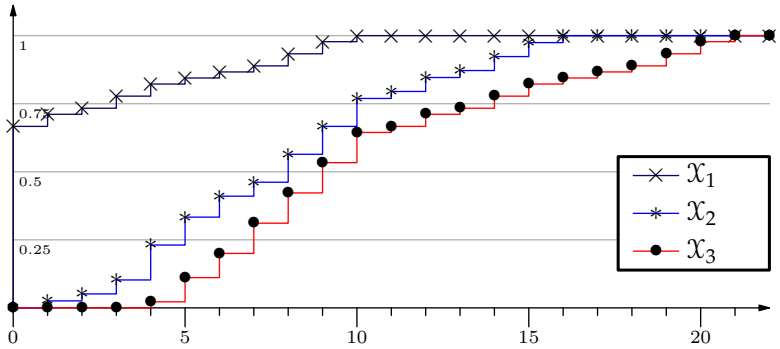
Definición 19. *Dado un conjunto de variables aleatorias $\{\mathcal{X}_i\}$ y un conjunto de "pesos" $\{p_i\}$, que cumplan $\sum_i p_i \leq 1$, definimos la **ponderación** del conjunto $\{\mathcal{X}_i\}$ con respecto a los pesos $\{p_i\}$, como la variable aleatoria $\bar{\mathcal{X}}$ cuya función de probabilidad viene dada por:*

$$f_{\bar{\mathcal{X}}}(\cdot) = \sum_i p_i f_{\mathcal{X}_i} \tag{5.1}$$

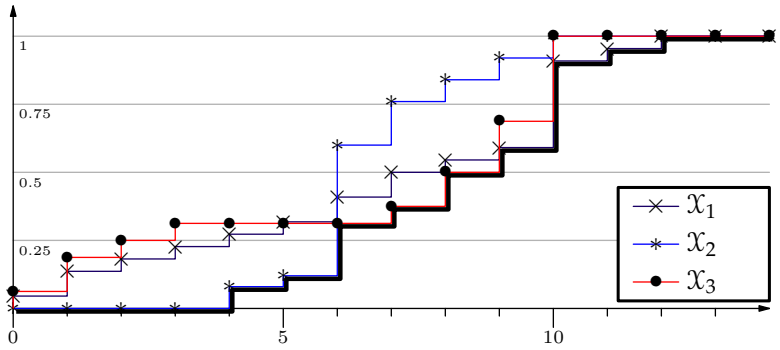
Proposición 6. *Si $\mathcal{X} \succcurlyeq \mathcal{X}_i$ para todo i, entonces $\mathcal{X} \succcurlyeq \bar{\mathcal{X}}$, siendo $\bar{\mathcal{X}}$ una ponderación cualquiera del conjunto $\{\mathcal{X}_i\}$.*

Demostración. De la definición de ponderación se deduce una expresión análoga para las funciones de probabilidad acumuladas:

$$F_{\bar{\mathcal{X}}}(\cdot) = \sum_i p_i F_{\mathcal{X}_i}(\cdot)$$



(a) En este caso $\mathcal{X}_3 = \sup_i\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3\}$, ya que $\mathcal{X}_3$ es peor que cualquiera de ellas



(b) En este caso ninguna de las $\mathcal{X}_i$ es peor que las otras. El supremo se construye artificialmente como la variable aleatoria cuya función de probabilidad acumulada es la dibujada en trazo grueso

Figura 5.6.: Supremo de un conjunto de variables aleatorias

Por hipótesis $\mathcal{X} \succcurlyeq \mathcal{X}_i$, luego $F_{\mathcal{X}}(\cdot) \leq F_{\mathcal{X}_i}$, y llevando esto al sumatorio anterior:

$$F_{\bar{\mathcal{X}}}(\cdot) \leq \sum_i p_i F_{\mathcal{X}}(\cdot) = F_{\mathcal{X}}(\cdot) \sum_i p_i$$

y ya que los pesos p_i deben cumplir que su sumatorio sea menor o igual que uno, se sigue que $F_{\bar{\mathcal{X}}}(\cdot) \leq F_{\mathcal{X}}(\cdot)$, es decir, $\mathcal{X} \succcurlyeq \bar{\mathcal{X}}$. □

Esta propiedad es útil, por ejemplo, para aproximar el tiempo de ejecución de una tarea por el supremo de sus tiempos, en lugar de usar el promedio de todos ellos. También será útil en la demostración de otros teoremas.

5.1.5. Mínimo de un conjunto de variables aleatorias

Siguiendo la misma idea de la sección anterior, podemos definir el mínimo (o ínfimo) de un conjunto de variables aleatorias como aquella cuya función de probabilidad acumulada está siempre *por encima* de las de las demás. Aunque en principio este concepto parece menos útil de cara a un análisis conservador, veremos en la siguiente sección que resultará útil para aproximar tiempos de bloqueo. Por tanto lo definiremos también formalmente a continuación.

Definición 20. *Dado un conjunto de variables aleatorias* $\{\mathcal{X}_i\}$*, llamaremos "ínfimo" de ese conjunto y lo denotaremos por* $\inf_i\{\mathcal{X}_i\}$ *a la variable aleatoria cuya función de distribución se construye en la forma siguiente:*

$$F_{\inf_i\{\mathcal{X}_i\}}(x) = \max_i F_{\mathcal{X}_i}(x)$$

Esta variable aleatoria, por construcción cumple que:

$$\mathcal{X}_i \succcurlyeq \inf_i\{\mathcal{X}_i\} \quad \forall i$$

5.2. Bloqueo

5.2.1. Introducción al problema y análisis clásico

El modelo de sistema enunciado en el capítulo 3, exigía que las tareas fuesen independientes. Esta suposición no es muy razonable, ya que en cualquier aplicación útil las tareas deberán comunicarse entre sí de alguna forma para cooperar. Un método habitual para esta comunicación es el acceso a recursos compartidos. Una tarea puede acceder por ejemplo a una posición de memoria para dejar un dato que otra tarea puede leer más adelante.

El problema del acceso a los recursos compartidos es que debe garantizarse que sean accedidos en forma exclusiva. Es decir, mientras una tarea está utilizando ese

recurso, ninguna otra debería poder usarlo. Esto se logra habitualmente mediante el uso de *semáforos*. Cuando una tarea requiere utilizar un recurso compartido, solicita al sistema operativo la posesión de un semáforo. Si lo obtiene, puede comenzar a utilizar el recurso. Cuando haya finalizado con él, liberará el semáforo. El operativo no dará el semáforo a ninguna otra tarea mientras tanto. El problema es que si la tarea solicita el semáforo, pero éste está en posesión de otra, la tarea debe *bloquearse*. En este estado, la tarea no compite por el procesador. Saldrá de este estado cuando el semáforo que necesitaba quede libre. Entonces podrá competir de nuevo por el procesador, y si es la de mayor prioridad, entrará en ejecución.

Esto introduce una nueva fuente de retardo (o interferencia) en la tarea. Además de poder ser expulsada por trabajos de mayor prioridad, la tarea puede quedar bloqueada por otras tareas (incluso de menor prioridad) cuando intenta acceder a un recurso compartido.

Peor aún, la implementación de los semáforos en el operativo puede causar lo que se denomina "inversión de prioridades". Cuando ocurre este problema, una tarea de alta prioridad puede verse interferida por una de baja prioridad incluso si no usan los mismos semáforos. Esto ocurre, por ejemplo, si una tarea de alta prioridad necesita un recurso que es poseído por una tarea de baja prioridad, y se bloquea. Durante el bloqueo, la tarea de baja prioridad puede ser expulsada de todas formas por tareas de prioridad intermedia, impidiéndole liberar el recurso. Durante este tiempo en que está expulsada, la tarea de alta prioridad está siendo interferida en realidad por la tarea de prioridad media. Ya que esta situación puede repetirse un número arbitrario de veces, el tiempo durante el cual puede permanecer bloqueada la tarea de alta prioridad no está acotado, lo que imposibilita el análisis desde un punto de vista de tiempo-real. Para una descripción detallada del problema véase [Buttazzo, 1997; Burns y Wellings, 2001].

En la literatura se han sugerido varias formas nuevas de implementar el bloqueo mediante semáforos, de forma que se evite el problema de la inversión de prioridades y de forma que el tiempo máximo que una tarea puede permanecer bloqueada esté acotado y sea calculable. Por ejemplo, el mecanismo de *Herencia de Prioridades* o PIP[1] [Sha *et al.*, 1990] hace que, si una tarea de alta prioridad necesita un recurso que está poseído por una tarea de baja prioridad, la tarea de baja hereda la prioridad alta mientras posea el semáforo, de modo que no pueda ser expulsada por otras de prioridad intermedia.

El mecanismo de *Techo de Prioridades* o CPP [2] [Sha *et al.*, 1990] hace que una tarea de alta prioridad que solicita un recurso por primera vez, lo obtenga sólo si puede garantizarse que encontrará libres todos los demás que va a necesitar. De este modo, si la tarea se bloquea lo hará sólo una vez al entrar en su primera región

[1] *Priority Inheritance Protocol*
[2] *Ceiling Priority Protocol*

crítica. En este protocolo resulta mucho más sencillo calcular el máximo tiempo de bloqueo que una tarea cualquiera puede experimentar. El inconveniente es que requiere que el programador de las tareas calcule de antemano para cada semáforo que utiliza una cifra llamada "Techo de prioridad de un semáforo", ya que el operativo necesita este dato para decidir si conceder o no un recurso a la tarea que lo pide.

El *Techo de Prioridad Inmediato* o ICPP [3] [Burns y Wellings, 2001] es una implementación más sencilla y directa de la idea anterior, en la que cada tarea se ejecuta con una prioridad igual al techo de prioridad de los semáforos que usa. Al igual que con el protocolo anterior, cada tarea se bloquea una vez como máximo, y la diferencia es que en este caso lo hace justo al principio de su ejecución (es decir, en realidad no entra en ejecución hasta que no se pueda garantizar que encontrará todos los recursos necesarios libres), en lugar de bloquearse al entrar en su primera región crítica. Desde el punto de vista del análisis de planificabilidad este protocolo es idéntico al anterior, pero desde el punto de vista de su implementación en un sistema operativo es más sencillo.

Finalmente, la *Política de Pila de Recursos*o SRP [4] [Baker, 1991] es una generalización de los métodos de techo de prioridades, que a diferencia de éstos, puede ser aplicada también con planificación EDF.

Cualquiera de estos métodos en el fondo es un mecanismo para garantizar que el máximo tiempo que una tarea puede estar bloqueada por otras de menor prioridad está acotado. Llamemos B_i al bloqueo máximo que puede experimentar la tarea τ_i. Cada política de implementación de los semáforos que hemos comentado da lugar a un B_i diferente, y proporciona diferentes métodos para calcular su valor. Los parámetros necesarios para obtener B_i suelen ser las duraciones máximas de las regiones críticas que cada tarea τ_j tiene, guardadas por el semáforo S_k. Denotaremos esta cantidad por $D_{j,k}$, que es la más frecuente en la literatura. Aunque la notación puede confundirse a primera vista con la del plazo, obsérvese que el plazo tiene un único subíndice. Además el contexto dejará claro cuándo se trata de una duración de sección crítica.

Sea cual sea la política y el método para obtener B_i, una vez calculado éste valor se añade a la fórmula del análisis clásico como si fuera parte del tiempo de ejecución de la tarea.

Esta forma de operar introduce pesimismo que proviene de varias fuentes diferentes:

1. Se asume que la tarea va a ser bloqueada todas las veces que pueda bloquearse. Por ejemplo, en la política CPP, la tarea sólo podrá bloquearse una vez, pero se asume que lo hará, cuando en la práctica habrá veces que no

[3] *Inmediate Ceiling Priority Protocol*
[4] *Stack Resource Policy*

se bloquee. Que se bloquee o no depende en realidad de si hay otras tareas accediendo al mismo recurso que ella necesita en ese momento, lo cual depende de los instantes de llegada de las restantes tareas.

2. Se asume que cuando una tarea se bloquea por culpa de otra, lo hace en el peor momento posible, esto es, justo cuando la otra acababa de entrar en su región crítica. Por tanto el bloqueo durará tanto como dure la región crítica de la otra.

3. Se asume que las regiones críticas de cada tarea duran un tiempo $D_{j,k}$ determinista. En la práctica el tiempo que una tarea requiera para completar su región crítica será una variable aleatoria, pero el análisis toma solamente su valor máximo.

Las dos primeras fuentes de pesimismo son muy difíciles de erradicar, ya que para tener en cuenta la "verdadera" probabilidad que existe de que una tarea se bloquee, necesitaríamos conocer no sólo las duraciones de las regiones críticas $D_{j,k}$, sino también en qué momento aparece cada una dentro del flujo de ejecución de cada tarea. Si además tenemos en cuenta que los tiempos de ejecución de los trabajos no son deterministas, vemos que el instante de entrar en una región crítica tampoco lo es. El análisis estocástico exacto de esta situación implica tener en cuenta todos los posibles escenarios de bloqueo y la probabilidad de cada uno. Esto produce una explosión combinatoria que hace el tiempo de análisis exponencial.

En cambio, la tercera fuente de pesimismo sí puede ser erradicada de una forma que apenas complica el análisis. Presentamos en esta sección una forma de incluir en el análisis los tiempos de las secciones críticas $\mathcal{D}_{j,k}$ como variables aleatorias, y de determinar a partir de sus distribuciones una distribución estadística para el tiempo de bloqueo de cada tarea, que garantizaremos "peor" que la verdadera. Añadiendo este tiempo de bloqueo al tiempo de ejecución de cada tarea, obtendremos una distribución del tiempo de respuesta pesimista, que tiene en cuenta el efecto del bloqueo.

5.2.2. Caso estocástico

En realidad el tiempo que un trabajo Γ_j puede estar bloqueado por otros de menor prioridad es una variable aleatoria que denotaremos por $\mathcal{B}_j$, que puede tomar cualquier valor entre cero y su máximo. De hecho, es frecuente que tome el valor cero, es decir, que la tarea no experimente bloqueo alguno porque no coincide que entre en su zona crítica mientras otra de menor prioridad tiene ocupado el recurso necesario. El análisis clásico se centra en obtener el valor máximo de $\mathcal{B}_j$, sin tener en cuenta su probabilidad.

Desde el punto de vista estocástico, sería deseable poder encontrar la función de probabilidad de $\mathcal{B}_j$, es decir, qué probabilidad hay de que el bloqueo sea de 0, 1, 2,... unidades de tiempo. Sin embargo, como ya se mencionó en la introducción, este problema resulta inabordable debido a que, por un lado, el modelo del sistema debería incluir información sobre el instante en que cada tarea entra en su región crítica, y por otro el análisis requeriría un tiempo que crecería exponencialmente con el número de tareas y secciones críticas. Además, la probabilidad de que la tarea sufra un bloqueo de b unidades, es diferente en cada instante del tiempo, por lo que en realidad $\mathcal{B}_j$ es un proceso estocástico que depende de t, de una forma muy compleja ya que las probabilidades de que sufra un nuevo bloqueo dependen de si ya lo ha sufrido antes.

Es evidente que un análisis exacto que incorpore el efecto del bloqueo resulta intratable. En su lugar usaremos una serie de simplificaciones, demostrando que en todo caso el tiempo de respuesta que obtenemos en el análisis es siempre "peor" que el verdadero, y que por tanto el análisis es conservador.

Estas son las simplificaciones:

1. Cada trabajo Γ_j sufrirá tantos bloqueos como permita el protocolo usado[5].

2. De cara al análisis, suponemos que todos estos bloqueos tienen lugar cuando el trabajo llega al sistema[6].

Estas dos simplificaciones producirán un tiempo de respuesta con una distribución "peor" que la que obtendríamos considerando la realidad. En la realidad, algunos de los bloqueos no tienen lugar en todas las activaciones del trabajo, sino sólo en algunas de ellas dependiendo de los patrones de llegadas de las otras tareas. Al asumir en nuestro análisis que siempre tendrán lugar, introducimos pesimismo. Por otro lado, en la realidad los instantes de bloqueo no están todos al principio de la tarea. Al moverlos a este punto estamos incrementando artificialmente la probabilidad de que el trabajo sea expulsado por otro de mayor prioridad que llegue después, y por tanto produciendo un tiempo de respuesta peor que el real.

Pero gracias a este par de simplificaciones podremos obtener el tiempo de respuesta que tiene en cuenta el bloqueo de una forma trivial:

$$\mathcal{R}_j^{\langle 0 \rangle} = \mathcal{W}_{P_j}(\lambda_j) + \mathcal{C}_j + \mathcal{B}_j$$

[5] Por ejemplo, bajo CPP el máximo de bloqueos es uno, mientras que bajo PIP es igual, bien al número de secciones críticas que tiene el trabajo, bien al número de trabajos de menor prioridad que podrían bloquearle, el que sea menor.

[6] Esto es cierto para el protocolo ICPP, pero no lo es para el CPP en el que el bloqueo sólo puede ocurrir cuando el trabajo entra en su primera sección crítica, ni para el PIP en el que el bloqueo puede ocurrir cada vez que el trabajo entra en una de sus secciones críticas.

Asumiendo que conocemos la función de probabilidad de $\mathcal{B}_j$, basta convolucionar ésta con la del tiempo de ejecución del trabajo, $\mathcal{C}_j$, y con la carga de prioridad P_j existente en λ_j para obtener una primera aproximación del tiempo de respuesta. A partir de este punto se continúa aplicando el análisis estocástico ya conocido, es decir, el método "partir, convolucionar y juntar" para cada uno de los trabajos posteriores de mayor prioridad que pueden expulsar a Γ_j.

Evidentemente queda el problema de calcular la función de probabilidad de $\mathcal{B}_j$. De nuevo un cálculo exacto no sería posible, pero sí podemos obtener una distribución de $\mathcal{B}_j$ que sea peor (garantizada) que la real. Esto es lo que haremos en las siguientes secciones, según la política de implementación de los semáforos.

5.2.3. Ampliación del modelo para incluir las duraciones de las secciones críticas

El modelo se amplía en la forma siguiente:

- El sistema se compone de N tareas $\{\tau_1, \tau_2, \ldots, \tau_N\}$ que pueden acceder a recursos compartidos, los cuales se guardan con K semáforos $\{S_1, S_2, \ldots, S_K\}$.
- Cada tarea se caracteriza por los parámetros ya contemplados en el modelo anterior, esto es: periodo, fase, tiempo de ejecución (aleatorio) y plazo.
- Para cada pareja tarea-semáforo, (τ_i, S_k), tenemos una variable aleatoria $\mathcal{D}_{i,k}$ que es la duración de la región crítica que tiene la tarea τ_i guardada por el semáforo S_k. Esta variable aleatoria viene definida por su función de probabilidad.

  Si la tarea τ_i no tiene ninguna región crítica guardada por el semáforo S_k, el correspondiente $\mathcal{D}_{i,k}$ será la variable aleatoria nula O de la definición 15.

  Si por el contrario la tarea τ_i tiene varias regiones críticas guardadas por el semáforo S_k, cada una con una duración diferente[7], tomaremos $\mathcal{D}_{i,k}$ como el supremo de estas duraciones, como se definió en la definición 18.

5.2.4. Función de probabilidad del bloqueo para el protocolo de herencia de prioridades

Este protocolo cumple un par de interesantes propiedades [Buttazzo, 1997], que enunciamos aquí de un modo más conveniente para nuestro análisis:

1. Un trabajo Γ_j no puede bloquearse dos veces a causa de un mismo semáforo.

[7] Esto es, con una distribución estadística diferente de la duración

2. Un trabajo Γ_j no puede ser bloqueado dos veces por un mismo trabajo de menor prioridad.

De estas dos propiedades, se deduce clásicamente que el número de bloqueos que Γ_j puede sufrir es mín(m,n), siendo m el número de semáforos que pueden bloquearle, y n el número de trabajos de menor prioridad que pueden bloquearle. Sin embargo, para conocer el tiempo máximo de bloqueo que puede experimentar, no basta con conocer n y m, sino que es necesario conocer la duración de cada una de las secciones críticas de otros trabajos que podrían causarle bloqueo y analizar todas las posibles combinaciones de bloqueos que pueden ocurrir, para tomar la peor.

Buttazzo [1997] propone un método menos costoso para encontrar, sino el caso peor, al menos uno que sea aún peor que el caso peor. El método consiste en calcular primero el tiempo de bloqueo que el trabajo experimentaría si todos los semáforos le bloquearan, y por otro lado el que sufriría si todos los trabajos de menor prioridad le bloquearan. Finalmente se queda con el menor de estos dos. Este método no es óptimo, ya que puede encontrar un valor para el bloqueo que nunca podría darse en la práctica porque violaría una de las condiciones antes enumeradas, pero garantiza que el valor obtenido es mayor o igual que el que se obtendría analizando exhaustivamente todos los posibles casos. Esta aproximación puede ser "traducida" al caso estocástico como veremos más adelante.

Algoritmo de búsqueda exhaustiva

Daremos a continuación un algoritmo que rastrea exhaustivamente todas las combinaciones de bloqueo "legales" (automáticamente excluye las que violarían las propiedades del protocolo PIP antes enunciadas). A medida que se produce el rastreo, se va actualizando una variable global que mantiene la peor distribución encontrada hasta el momento. Para simplificar, asumiremos que las tareas están numeradas por orden de prioridad, esto es, τ_1 es la de mayor prioridad y τ_N la menos prioritaria.

Al igual que Buttazzo [1997], denominaremos "techo de prioridad" de un semáforo S_k, y lo denotaremos por $C(S_k)$, al máximo de las prioridades de las tareas que tienen secciones críticas guardadas por el semáforo S_k. Es evidente que una tarea sólo podrá sufrir bloqueo por un semáforo S_k si el techo de prioridad del semáforo es mayor o igual que la prioridad de la tarea.

Nuestro algoritmo hace uso de unas marcas asociadas a cada semáforo que lo clasifican como "disponible" o "no-disponible"[8]. Sólo se tendrá en cuenta el bloqueo causado por una región crítica, si el semáforo que la guarda está marcado como "disponible". Inicialmente todos los semáforos que pueden causar bloqueo

[8] Estas marcas solo afectan al algoritmo para el cálculo de la distribución de $\mathcal{B}_i$, no necesitan ser implementadas en el sistema operativo

están disponibles. Estos son los semáforos con un techo de prioridad mayor o igual que la prioridad de la tarea cuyo $\mathcal{B}_i$ estamos buscando. A medida que se van considerando los bloqueos que cada tarea de menor prioridad puede causar, se marcan como "no-disponibles" los correspondientes semáforos. De este modo se evita considerar dos bloqueos causados por el mismo semáforo.

El cuerpo principal del algoritmo sería muy simple:

```
1: Calcular techo de prioridad C(S_k) para cada S_k
2: para i = 1,...,N hacer
3:    Marcar como "disponible" todo S_k tal que C(S_k) ≥ P_i
4:    B_i ← 0
5:    ACTUALIZARPEORBLOQUEO(i + 1, 0)
6: fin para
```

En este pseudocódigo nos referimos a $\mathcal{B}_i$ que es en realidad una variable aleatoria. En una implementación real operaremos en realidad con sus funciones de probabilidad o de probabilidad acumulada, según convenga.

Evidentemente el verdadero trabajo lo hace la rutina ACTUALIZARPEORBLOQUEO, cuyo pseudocódigo sería el mostrado en la figura 5.7.

Esta rutina recibe dos parámetros:

- El índice de la tarea por debajo de la cual (inclusive) debe calcularse el bloqueo. Así, para obtener $\mathcal{B}_i$ para la tarea τ_i, es necesario considerar todas las tareas de menor prioridad, de ahí el $(i+1)$. (Recordar que suponemos las tareas ordenadas por prioridades, siendo τ_1 la de mayor prioridad).

- El valor del bloqueo calculado hasta el momento. En la primera llamada es 0, pero a medida que la función se llama recursivamente a sí misma, va incrementándose.

La explicación del algoritmo de la figura 5.7 es como sigue. Si no estamos en la tarea de menor prioridad de todas (que es la tarea N-ésima), entonces se considera "qué pasaría si la tarea actual no entrara en la sección crítica", mediante la llamada recursiva de la línea 3, que se limita a calcular los bloqueos debidos a tareas inferiores, sin añadir nada a la variable *tmp* recibida. Posteriormente se considera el "qué pasaría si la tarea actual entrara en la sección guardada por el semáforo S_k", para cada S_k disponible, mediante la llamada de la línea 6. En ella se añade al tiempo de bloqueo temporal el debido a la región $\mathcal{D}_{j,k}$, y llama recursivamente para continuar con tareas de menor prioridad. Observar que cuando estamos mirando "qué pasaría al bloquearse en el semáforo S_k", marcamos este semáforo como no-disponible, de modo que ya no será considerado como posibilidad para las tareas de menor prioridad en la llamada recursiva. Con este método

```
1: ACTUALIZARPEORBLOQUEO(j, tmp)
2: si j < N entonces
3:    ACTUALIZARPEORBLOQUEO(j + 1, tmp)
4:    para todo k tal que S_k esté disponible hacer
5:       Marcar S_k como no-disponible
6:       ACTUALIZARPEORBLOQUEO(j + 1, tmp + D_{j,k})
7:       Marcar S_k como disponible
8:    fin para
9: si no
10:   Bloqueo← tmp + sup{D_{j,k} | S_k disponible}
11:   B_i ← sup{Bloqueo, B_i}
12: fin si
```

Figura 5.7.: Algoritmo recursivo para encontrar el peor caso de bloqueo, bajo el protocolo de herencia de prioridades

garantizamos que cada semáforo no causará más de un bloqueo. Notar que la suma $tmp + \mathcal{D}_{j,k}$ es en realidad una suma de variables aleatorias, por lo que lo que habría que realizar en este punto es la convolución de sus funciones de probabilidad.

Eventualmente la recursividad alcanzará la tarea de menor prioridad τ_N. En este caso, añadimos al bloqueo temporal que teníamos hasta entonces el máximo bloqueo que podría causar esta tarea de menor prioridad, es decir, el máximo (supremo) entre los $\mathcal{D}_{N,k}$ para aquellos k tales que S_k no hayan sido marcados como ocupados (línea 10). Para cubrir el caso en que ya todos los semáforos están marcados, asumiremos por convenio que el supremo de un conjunto vacío es la variable aleatoria nula $\mathbb{O}$. De nuevo esta suma es de variables aleatorias, por lo que lo que se realiza es la convolución de sus funciones de probabilidad. La variable aleatoria así obtenida, se compara con $\mathcal{B}_i$ para quedarse con la peor (supremo) de ambas, y actualizar $\mathcal{B}_i$ con el resultado, cosa que se realiza en la línea 11.

Observar que la variable $\mathcal{B}_i$ se va actualizando cada vez que la recursividad llega a la tarea de menor prioridad. Al final, $\mathcal{B}_i$ contendrá la peor distribución de entre todas las posibles. Este método es también aplicable al caso determinista, sustituyendo los supremos por máximos.

Ejemplo

Plantearemos el mismo ejemplo que el resuelto en [Buttazzo, 1997], en el que las duraciones de las secciones críticas son deterministas. Más adelante daremos otro ejemplo con duraciones aleatorias.

El cuadro 5.1 muestra la duración de cada región crítica para cada tarea y cada

semáforo. Un guión indica que la tarea en cuestión no tiene regiones guardadas por ese semáforo. Nos planteamos calcular el peor bloqueo que puede sufrir la tarea de mayor prioridad τ_1, aplicando el algoritmo recursivo antes visto.

	S_1	S_2	S_3
τ_1	1	2	–
τ_2	–	9	3
τ_3	8	7	–
τ_4	6	5	4

Cuadro 5.1.: Ejemplo de cálculo de B_i, con duraciones deterministas [Buttazzo, 1997]

En primer lugar, el techo de prioridad de los semáforos (definido como la prioridad más alta entre la de los trabajos que usan ese semáforo) será P_1 para S_1 y S_2 (ya que ambos semáforos son usados por τ_1) y P_2 para S_3. Por tanto a la hora de calcular el bloqueo sufrido por τ_1, sólo se deben tener en cuenta S_1 y S_2, pero no S_3.

El algoritmo comienza pues marcando como disponibles S_1 y S_2, inicializando el peor bloqueo B_1 con cero, y haciendo la llamada **ActualizarPeorBloqueo**(2,0). Llamaremos "nivel" al primer parámetro. A partir de aquí, la secuencia de llamadas recursivas transcurre en la forma siguiente:

- Nivel 2, llamamos recursivamente para el siguiente nivel, pasando el mismo $tmp = 0$.
  - Nivel 3, se llama recursivamente para el siguiente nivel, pasando el mismo $tmp = 0$.
    - Nivel 4. estamos en la tarea más baja. Ya no hay llamada recursiva. Se suma a *tmp* la duración más larga de las regiones críticas "disponibles". Tenemos disponibles ambos semáforos S_1 y S_2, por lo que tomamos 6 que es la correspondiente a S_1. El bloqueo de esta combinación es por tanto 6, y actualizamos B_1 con este valor.
  - De nuevo en nivel 3, consideramos ahora cada uno de los semáforos que tenemos libres a este nivel. Primero S_1. Lo marcamos como no disponible, y añadimos 8 (duración de la región crítica guardada por S_1) a *tmp*. Llamamos recursivamente
    - Nivel 4, ahora solo tenemos disponible S_2. El máximo bloqueo que podemos causar será por tanto 5, lo que sumamos a *tmp*. El bloqueo de esta combinación es 13. Actualizamos B_i
  - Y ahora probamos usando el otro semáforo S_2. En este caso *tmp* se incrementa en 7. Llamamos recursivamente

        - Nivel 4, ahora solo tenemos disponible S_2. El máximo bloqueo que esta tarea puede causar es 6, lo que se suma al *tmp* recibido. El bloqueo de esta combinación es 13. No es peor que el B_i almacenado, luego éste no se actualiza.
    - Agotadas todas las posibilidades en el nivel 3, retornamos.

- De nuevo en nivel 2, consideramos ahora cada uno de los semáforos que tenemos libres a este nivel. Tenemos disponibles S_1 y S_2, pero la tarea τ_2 sólo usa S_2, de modo que ésta es la única posibilidad. Lo marcamos como no-disponible y sumamos a *tmp* (que era cero en este nivel) el valor 9 (duración de la región crítica asociada a S_2). Llamamos recursivamente.
    - Nivel 3, ahora con $tmp = 9$ y con S_2 no disponible. La primera llamada recursiva no usa ningún semáforo adicional ni incrementa *tmp*:
        - Nivel 4, tenemos disponible S_1, por tanto el peor bloqueo que esta tarea puede causar es 6, que se suma a *tmp* y produce 15. Actualizamos B_i con este valor.
    - De nuevo en nivel 3, consideramos ahora cada uno de los semáforos que tenemos libres a este nivel. Sólo está libre S_1. Lo marcamos como no disponible y añadimos 8 *tmp*. Llamamos recursivamente
        - Nivel 4, encontramos todos los semáforos no-disponibles. Esta tarea no puede causar bloqueo adicional en esta combinación. No se añade nada a *tmp* que resulta ser 17. Se actualiza B_i con este valor
    - Nivel 3, finalizadas todas las posibilidades, retornamos
- Nivel 2, finalizadas todas las posibilidades, retornamos

Al finalizar el algoritmo y la cadena de llamadas recursivas, vemos que tenemos finalmente en B_i la cantidad 17, que es por tanto el peor bloqueo que puede sufrir la tarea τ_1.

Aproximación para evitar la búsqueda exhaustiva

La simplificación propuesta por Buttazzo [1997] consiste en:

- Para cada tarea que pueda bloquear a τ_1 tomar su sección crítica más larga (independientemente de qué semáforo la guarde), y sumar todas estas cantidades. En el ejemplo saldría: $\text{máx}(9,3) + \text{máx}(8,7) + \text{máx}(6,5,4) = 9 + 8 + 6 = 23$. Es decir, se suma el máximo de cada fila (tras eliminar la columna correspondiente a S_3, cuyo techo de prioridad es menor de P_1). El resultado es denotado por B_1^l.

- Para cada semáforo que pueda bloquear a τ_1 (esto es, aquellos con $C(S_k) \geq P_1$), tomar su sección crítica más larga entre todos los trabajos de menor prioridad y sumar todas estas cantidades. En el ejemplo saldría: $\text{máx}(8,6) + \text{máx}(9,7,5) = 17$. Es decir, se suma el máximo de cada columna (tras eliminar la columna S_3 como se ha dicho). El resultado es denotado por B_1^s

- Tomar la menor de las dos cantidades anteriores, como aproximación pesimista del tiempo de bloqueo. Es decir $B_1 \leq \text{mín}(B_1^l, B_1^s)$. En el ejemplo sale 17, que en este caso coincide con lo obtenido por búsqueda exhaustiva.

Vemos que, para el ejemplo desarrollado antes, obtendremos exactamente la misma respuesta por búsqueda exhaustiva que con la aproximación. En general esto no será así, como se puede mostrar mediante el siguiente contraejemplo:

	S_1	S_2
τ_1	1	1
τ_2	2	3
τ_3	4	5

Para calcular B_1 en este sencillo sistema, si aplicamos la simplificación de Buttazzo [1997] obtendremos: sumando las peores secciones críticas de cada trabajo (máximos por filas): 3+5=8=B_1^l; sumando las peores secciones críticas de cada semáforo (máximos por columnas): 4+5=9=B_1^s. Tomando la menor de ambas sale 8. Sin embargo, este tiempo de bloqueo sólo puede aparecer si la tarea τ_2 causa bloqueo a través del semáforo S_2 y la tarea τ_3 también. Esta combinación no puede ocurrir nunca en la práctica, ya que un mismo semáforo (S_2) estaría causando dos bloqueos, lo que viola una de las propiedades del protocolo. La búsqueda exhaustiva localizaría correctamente el peor caso posible, de 7 unidades, que aparece cuando τ_2 causa bloqueo a través de S_2 y τ_3 a través de S_1.

En el caso determinista, la simplificación de Buttazzo [1997], encuentra un bloqueo que no es exacto, pero que al menos es mayor que el exacto, por lo que es una aproximación "segura" o conservadora. Cabe preguntarse si esta aproximación sigue siendo segura cuando se aplica al caso estocástico. La traducción de este algoritmo al lenguaje estocástico sería la siguiente:

- Para cada tarea τ_j de menor prioridad que τ_i, tomar el supremo de $\mathcal{D}_{j,k}$, para aquellos k tales que $C(S_k) \geq P_i$. Es decir, construimos una distribución estadística "peor" que la de cualquiera de sus secciones críticas.

  Sumar (convolucionar) todas las obtenidas. Llámese al resultado $\mathcal{B}_i^l$.

- Para cada semáforo S_k tal que $C(S_k) \geq P_i$, tomar el supremo de las $\mathcal{D}_{j,k}$ guardadas por ese semáforo, en tareas τ_j de menor prioridad que P_i

  Sumar (convolucionar) todas las obtenidas. Llámese al resultado $\mathcal{B}_i^s$

- Elegir el ínfimo entre $\mathcal{B}_i^l$ y $\mathcal{B}_i^s$.

Comenzaremos mostrando un ejemplo en el que veremos que efectivamente este método produce una distribución que, aunque no es exacta, sí es "peor" que la que se obtiene por búsqueda exhaustiva. Seguidamente demostraremos que esto ocurrirá siempre, para cualquier sistema, y que por tanto ésta es una aproximación segura.

Usaremos el mismo contraejemplo anterior, pero haciendo que algunas de las secciones críticas tengan duraciones aleatorias, en lugar de deterministas. El cuadro 5.2 muestra estas nuevas duraciones. Hemos hecho que las duraciones $\mathcal{D}_{2,2}$ y $\mathcal{D}_{3,2}$ sean ahora variables aleatorias, uniformemente distribuidas entre 1 y su valor máximo, que es de 3 y 5 respectivamente. Los restantes valores son deterministas, la notación D[2], por ejemplo, indica representa una variable "aleatoria" que sólo puede tomar el valor 2.

	S_1	S_2
τ_1	D[1]	D[1]
τ_2	D[2]	U[1,3]
τ_3	D[4]	U[1,5]

Cuadro 5.2.: Ejemplo que muestra que la simplificación de Buttazzo [1997] también es aplicable al caso estocástico

En este caso el método de búsqueda exhaustiva recorrerá todas las posibles combinaciones de bloqueos legales que se pueden plantear y que son:

1. La tarea τ_2 no causa bloqueo, y la tarea τ_3 sí. El "peor" bloqueo en este caso sería el supremo entre D[4] y U[1,5].
2. La tarea τ_2 se bloquea en S_1 y la tarea τ_3 en S_2. El bloqueo en este caso sería D[2]$\otimes$U[1,5].
3. La tarea τ_2 se bloquea en S_2 y la tarea τ_3 en S_1. El bloqueo en este caso sería U[1,3]$\otimes$D[4]

El resultado de la búsqueda exhaustiva será el supremo entre todos los anteriores, es decir:

$$\mathcal{B}_1 = \sup\{D[4], U[1,5], D[2] \otimes U[1,5], U[1,3] \otimes D[4]\}$$

La figura 5.8 muestra las CDFs de estos cuatro casos y puede observarse que el supremo de ellas coincide con U[1,3]$\otimes$D[4]. Esta sería por tanto la distribución del bloqueo $\mathcal{B}_1$ que buscábamos, que garantiza un análisis conservador.

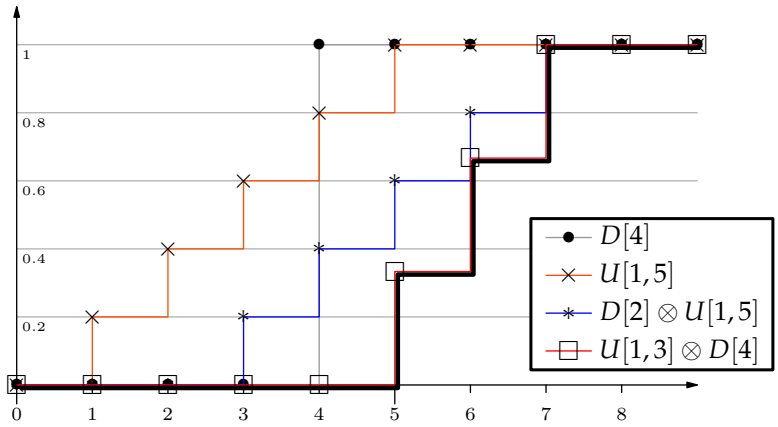


Figura 5.8.: Solución del contraejemplo

Si en cambio intentamos usar la aproximación que evita la búsqueda exhaustiva, los pasos a realizar serían:

- Para cada tarea τ_2, τ_3, encontrar el supremo de sus $\mathcal{D}_{j,k}$. Así, para τ_2 sería $\sup\{D[2], U[1,3]\}$. El resultado se muestra en la figura 5.9(a). Por otro lado, para τ_3 sería $\sup\{D[4], U[1,5]\}$, y el resultado se muestra en la figura 5.9(b).

  Finalmente, se suman (convolucionan) las distribuciones obtenidas. El resultado se muestra en la figura 5.9(c), etiquetado como $\mathcal{B}_1^l$.

- De forma análoga, se procede para cada semáforo S_1, S_2, encontrando el supremo de sus $\mathcal{D}_{j,k}$ y sumándolos (convolucionándolos). Es decir:

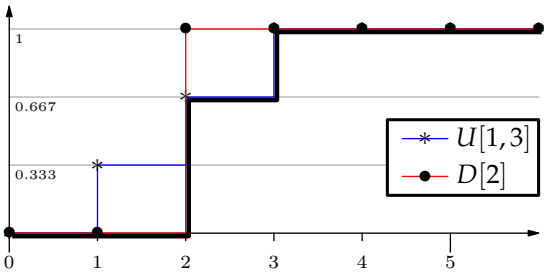
$$\mathcal{B}_1^s = \sup\{D[3], D[4]\} \otimes \sup\{U[1,3], U[1,5]\} = D[4] \otimes U[1,5]$$

  La figura 5.9(c) muestra también el resultado de esta operación.

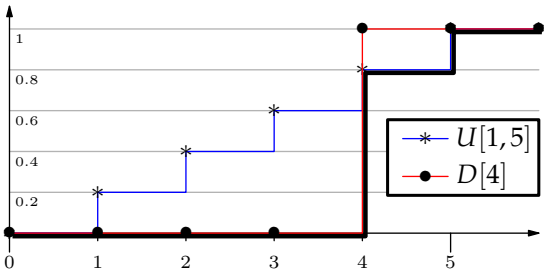
- Finalmente se elige el ínfimo de los dos, que se representa en la figura 5.9(c) en línea gruesa.

Si comparamos el resultado obtenido por aproximación y por búsqueda exhaustiva, vemos que son diferentes. La figura 5.10 muestra uno al lado del otro. Lo interesante es que la solución obtenida por aproximación está en todo momento por debajo de la obtenida por búsqueda exhaustiva, es decir, es "peor" en el sentido estocástico definido en este capítulo. En el siguiente teorema se demuestra que esto es así para cualquier sistema.

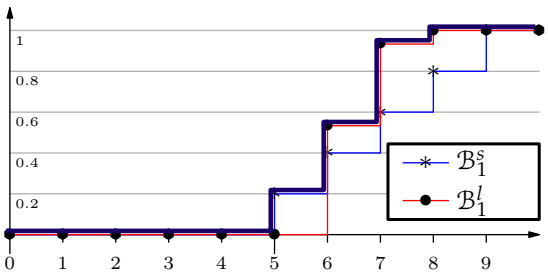
Teorema 9. *Llamemos $\mathcal{B}_i^E$ a la variable aleatoria que representa el bloqueo que puede experimentar una tarea τ_i, obtenido mediante el algoritmo de búsqueda exhaustiva, y $\mathcal{B}_i^A$ al obtenido mediante la aproximación antes descrita. Para cualquier sistema se cumple que $\mathcal{B}_i^A \succcurlyeq \mathcal{B}_i^E$.*



(a) Supremo del conjunto $\{D[2], U[1,3]\}$ (en línea gruesa)



(b) Supremo del conjunto $\{D[4], U[1,5]\}$ (en línea gruesa)



(c) Comparación entre $\mathcal{B}_1^l$ y $\mathcal{B}_1^s$

Figura 5.9.: Solución del contraejemplo por aproximación [Buttazzo, 1997]

Demostración. Para calcular el bloqueo de la tarea τ_i es necesario considerar las restantes tareas tareas de menor prioridad, $\tau_{i+1}, \tau_{i+2}, \ldots, \tau_N$. Cada una de estas tareas tiene k secciones críticas que pueden, al menos en principio, causar bloqueo a τ_i.

El algoritmo de búsqueda exhaustiva, en cada iteración, considerará un caso de interbloqueo que a lo sumo toma una sección crítica de cada una de estas tareas, puesto que respeta la propiedad de que una tarea no puede causar más de un bloqueo. Sea G el conjunto de las secciones críticas que se están considerando en una iteración de la búsqueda exhaustiva. En el conjunto G no habrá dos secciones críticas pertenecientes a la misma tarea, ni al mismo semáforo, por lo que en G habrá mín(N, K) secciones críticas, siendo N el número de tareas y K el número de semáforos.

Un ejemplo ayudará a comprender el desarrollo que sigue. La figura 5.11 muestra una tabla en la que se organizan las duraciones de las secciones críticas en un sistema con 5 tareas y 3 semáforos. Supongamos que se está buscando el bloqueo de la tarea 1, para lo cual hay que considerar las diferentes combinaciones de las secciones críticas de las restantes tareas de menor prioridad, $\tau_2, \ldots, \tau_5$. En

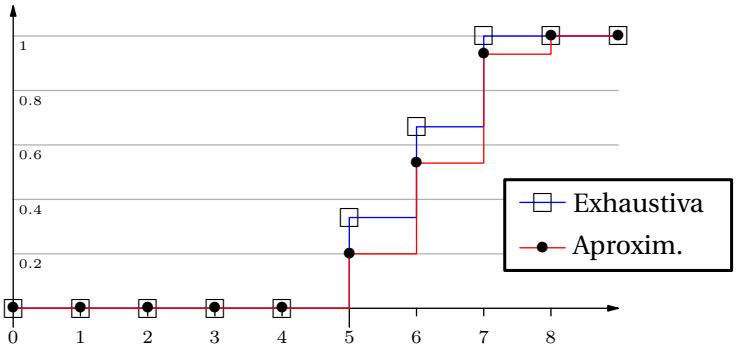


Figura 5.10.: Comparación entre la solución aproximada y la encontrada por búsqueda exhaustiva

un instante dado el algoritmo de búsqueda exhaustiva está considerando el caso marcado con círculos en la figura. Como vemos, se considera la primera sección crítica de la tarea τ_2, la segunda sección crítica de τ_5 y la tercera sección crítica de τ_3. Para la tarea τ_4 no se considera sección crítica alguna, pues ello implicaría que un semáforo estaría siendo considerado dos veces. Así pues, el conjunto G de secciones críticas que se está examinando en el instante recogido en la figura es $G = \{\mathcal{D}_{2,1}, \mathcal{D}_{2,3}, \mathcal{D}_{5,2}\}$.

	S_1	S_2	S_3
τ_1	$\mathcal{D}_{1,1}$	$\mathcal{D}_{1,2}$	$\mathcal{D}_{1,3}$
τ_2	$\mathcal{D}_{2,1}$	$\mathcal{D}_{2,2}$	$\mathcal{D}_{2,3}$
τ_3	$\mathcal{D}_{3,1}$	$\mathcal{D}_{3,2}$	$\mathcal{D}_{3,3}$
τ_4	$\mathcal{D}_{4,1}$	$\mathcal{D}_{4,2}$	$\mathcal{D}_{4,3}$
τ_5	$\mathcal{D}_{5,1}$	$\mathcal{D}_{5,2}$	$\mathcal{D}_{5,3}$

Figura 5.11.: Ilustración para la demostración del teorema 9

Consideremos uno cualquiera de los conjuntos G generados durante la búsqueda exhaustiva. Llamemos k_j al índice de la sección crítica con la que la tarea τ_j contribuye al conjunto G. Si la tarea no contribuye, $k_j = 0$. Así, en el ejemplo anterior tendremos que $k_2 = 1, k_3 = 3, k_4 = 0$ y $k_5 = 2$. El bloqueo que generaría la situación descrita por este conjunto G, que denotaremos por $\mathcal{B}_i(G)$, sería por consiguiente la suma de estas secciones críticas, una por fila:

$$\mathcal{B}_i(G) = \sum_{j=i+1}^{N} \mathcal{D}_{j,k_j}$$

asumiendo que una sección crítica de la forma $\mathcal{D}_{*,0}$ (esto es, con un cero en su segundo subíndice) tiene una duración nula, o si se prefiere, igual a la variable aleatoria $\mathbb{O}$. En el caso del ejemplo, el conjunto G marcado con círculos en la figura genera un bloqueo igual a $\mathcal{D}_{2,1} + \mathcal{D}_{3,3} + \mathcal{D}_{4,0} + \mathcal{D}_{5,2}$. Observar que $\mathcal{D}_{4,0} = \mathbb{O}$.

El algoritmo de búsqueda exhaustiva calculará el tiempo de bloqueo como el "peor" (supremo) de todos los obtenidos para cada uno de los posibles conjuntos G, es decir:

$$\mathcal{B}_i^E = \sup_G \{\mathcal{B}_i(G)\}$$

El algoritmo aproximado, en cambio, computará una aproximación "por tareas" (o si se prefiere, por filas), denotada por $\mathcal{B}_i^l$ y otra "por semáforos" (o si se prefiere, por columnas), denotada por $\mathcal{B}_i^s$, para finalmente quedarse con la "mejor" (ínfimo) de las dos.

Nos centraremos de momento en el cálculo por tareas o filas. Esta aproximación consiste en sumar (convolucionar) la distribución peor de cada fila, por debajo de la fila i, es decir:

$$\mathcal{B}_i^l = \sum_{j=i+1}^{N} \sup_k \{\mathcal{D}_{j,k}\}$$

Si comparamos esta aproximación con el valor de $\mathcal{B}_i(G)$ obtenido para un conjunto G en la búsqueda exhaustiva, encontraremos que, sea cual sea G, la aproximación por filas es "peor". En efecto, en la aproximación cada término del sumatorio es el supremo de una fila, y este supremo es "peor" por definición que cualquiera de los elementos de la fila, en particular, peor que el elemento que contribuye a la solución por búsqueda exhaustiva. Es decir:

$$\sup_k \{\mathcal{D}_{j,k}\} \succcurlyeq \mathcal{D}_{j,k_j}$$

Esta afirmación es cierta también si la fila (tarea) j no contribuye al conjunto G, ya que en este caso $k_j = 0$ y por tanto $\mathcal{D}_{j,k_j} = \mathbb{O}$; y es evidente que $\sup_k\{\mathcal{D}_{j,k}\} \succcurlyeq \mathbb{O}$. Por tanto podemos asegurar que siempre

$$\mathcal{B}_i^l = \sum_{j=i+1}^{N} \sup_k \{\mathcal{D}_{j,k}\} \succcurlyeq \sum_{j=i+1}^{N} \mathcal{D}_{j,k_j} = \mathcal{B}_i(G) \qquad \text{sea cual sea } G$$

De donde, $\mathcal{B}_i^l \succcurlyeq \mathcal{B}_i(G)$, es decir, la aproximación "por filas" (o por tareas) es peor que la solución hallada en cualquiera de los pasos de la búsqueda exhaustiva. Por tanto también será peor que el supremo de todos los $\mathcal{B}_i(G)$, que es el resultado final de la búsqueda exhaustiva. Por tanto $\mathcal{B}_i^l \succcurlyeq \mathcal{B}_i^E$.

Un razonamiento totalmente análogo nos lleva a la conclusión de que la aproximación "por columnas" (o por semáforos) es también peor que la solución hallada por búsqueda exhaustiva, es decir $\mathcal{B}_i^s \succcurlyeq \mathcal{B}_i^E$.

Comparando las funciones de probabilidad acumuladas, tenemos pues que, para cualquier valor de x se cumplirá que

$$F_{\mathcal{B}_i^s}(x) \leq F_{\mathcal{B}_i^E}(x)$$
$$F_{\mathcal{B}_i^l}(x) \leq F_{\mathcal{B}_i^E}(x)$$

De modo que aún quedándose con el máximo, punto a punto, de $F_{\mathcal{B}_i^s}$ y $F_{\mathcal{B}_i^l}$ la función que se obtiene es menor o igual que $F_{\mathcal{B}_i^E}$, para cada uno de los puntos. Y ya que la función que, punto a punto, se queda con el máximo de $F_{\mathcal{B}_i^s}$ y $F_{\mathcal{B}_i^l}$ corresponde a $\mathcal{B}_i^A$, se deduce que $\mathcal{B}_i^A \succcurlyeq \mathcal{B}_i^E$, como se quería demostrar. □

5.2.5. Función de probabilidad del bloqueo para los protocolos de techo de prioridad

Los protocolos de techo de prioridad, tanto el "original" como el "inmediato" (que es una elaboración posterior) garantizan que cada trabajo se bloqueará una vez como máximo. El análisis clásico busca la región crítica más larga de entre las que podrían causar bloqueo a la tarea τ_i. Su longitud es el valor de B_i, el bloqueo máximo que τ_i puede sufrir.

La "traducción" al caso estocástico es directa, sustituyendo el concepto de "máximo" por el de "supremo" establecido en la definición 18. Un algoritmo directo sería el siguiente:

1: Calcular el techo de prioridad $C(S_k)$ para cada semáforo S_k
2: **para** $i = 1, \ldots, N$ **hacer**
3: $\quad \mathcal{B}_i \leftarrow \mathbb{O}$
4: $\quad$ **para** $j = i+1, \ldots, N$ **hacer**
5: $\quad\quad$ **para** $k = 1, \ldots, K$ **hacer**
6: $\quad\quad\quad$ **si** $C(S_k) \geq P_i$ **entonces**
7: $\quad\quad\quad\quad \mathcal{B}_i \leftarrow \sup\{\mathcal{B}_i, \mathcal{D}_{j,k}\}$
8: $\quad\quad\quad$ **fin si**
9: $\quad\quad$ **fin para**
10: $\quad$ **fin para**
11: **fin para**

Es evidente que este algoritmo es mucho menos costoso que el presentado para la búsqueda exhaustiva bajo PIP. Desde el punto de vista del análisis, los mecanismos de techo de prioridades son pues preferibles. Además, el propio funcionamiento de este protocolo garantiza que, si un trabajo ha de sufrir bloqueo, lo sufrirá al principio de su ejecución (como una especie de retraso añadido). Esto

coincide con nuestra hipótesis 2 de la página 145, por lo que en este caso la hipótesis no incrementa el pesimismo, y por tanto los resultados de nuestro análisis serán más cercanos a la solución "real".

5.3. Jitter

Una segunda restricción del modelo del capítulo 3 es que los instantes de llegada de los trabajos son deterministas. La mayoría de los sistemas de tiempo real se componen de bucles de control, que deben ser ejecutados periódicamente cada cierto tiempo. En este sentido sí es cierto que los instantes de llegada deberían ser deterministas, al menos para este tipo de tareas.

Sin embargo, la implementación real hace que en ocasiones el instante en que el trabajo se activa esté ligeramente retrasado con respecto al instante teórico en que debería haberse activado (debido a imprecisiones del planificador, por ejemplo). La diferencia entre el instante de llegada teórico y el real y es lo que se denomina *jitter*. Una llegada retrasada de un trabajo puede hacer que éste cause interferencia a otros a los que no podría haber causado interferencia de haber llegado en el instante teórico. Es decir, pueden aparecer condiciones en la realidad que no habían sido previstas en el análisis. Como consecuencia las probabilidades reales de perder los plazos se incrementan con respecto a las obtenidas en el análisis.

Nos planteamos en esta sección modificar el análisis de modo que los las PF que se obtengan para el tiempo de respuesta sean peores que las reales, incluso teniendo en cuenta que en la realidad cada trabajo puede sufrir un *jitter* aleatorio (aunque acotado).

5.3.1. Ampliación al modelo para incluir el jitter

Cada tarea τ_i del sistema lleva un parámetro adicional, $\mathcal{J}_i$ que representa el *jitter*. Este es una variable aleatoria que puede tomar valores negativos (lo que indicaría que el trabajo puede adelantarse con respecto a su instante de llegada teórico) o positivos (lo que indicaría que puede retrasarse). Caracterizaremos esta variable aleatoria únicamente por sus valores extremos $J_i^{\min}$ y $J_i^{\max}$. El análisis no hará uso de otra información estadística sobre ella (es decir, no se requiere la PF del *jitter*, tan sólo sus valores extremos).

El problema que se nos plantea por tanto es el de determinar la PF de los tiempos de respuesta de un sistema S en el cual el trabajo $\Gamma_{j,i}$ (que es la j-ésima activación de la tarea τ_i) puede llegar en un instante aleatorio dado por:

$$\lambda_{i,j} = \Phi_i + T_i(j-1) + \mathcal{J}_i$$

siendo $\mathcal{J}_i$ una variable aleatoria de la que desconocemos su distribución.

5.3.2. Modificación del análisis para incluir el jitter

Lo que haremos será transformar el problema en otro, en el que los instantes de llegada sean deterministas, para poder aplicarle el mismo análisis ya visto en el capítulo 4. La transformación será tal que garantice que la solución del problema transformado es más pesimista que la del problema original.

La transformación es como sigue. Dado el sistema $S = \{\tau_i\}$ con N tareas, en el que cada tarea se especifica mediante la tupla $\tau_i = (\Phi_i, T_i, D_i, \mathcal{C}_i, \mathcal{J}_i)$ (opcionalmente también pueden incluirse los parámetros de bloqueo vistos en la sección 5.2.3), construimos el sistema simplificado $S' = \{\tau_i'\}$, en el que cada tarea τ_i' viene definida por la tupla:

$$\begin{cases} \Phi_i' = \Phi_i + J_i^{\min} \\ T_i' = T_i \\ D_i' = D_i \\ \mathcal{C}_i' = \mathcal{C}_i + (J_i^{\max} - J_i^{\min}) \end{cases} \tag{5.2}$$

(Los parámetros relacionados con el bloqueo, si los hubiera, serían los mismos para el sistema S' que para el S).

Es decir, el nuevo sistema S' es idéntico al S, excepto en los tiempos de ejecución de las tareas, que se han incrementado en una cantidad $(J_i^{\max} - J_i^{\min})$, y en que los instantes de llegada de los trabajos ya no son aleatorios, sino deterministas y ocurren en:

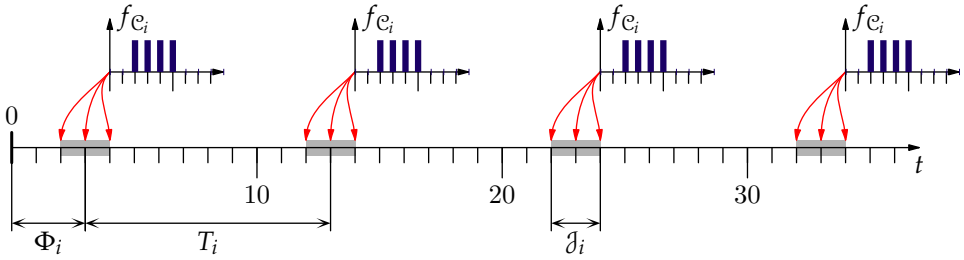
$$\lambda_{i,j}' = \Phi_i + J_i^{\min} + T_i(j-1) = \lambda_{i,j} + J_i^{\min} \tag{5.3}$$

Dicho de otro modo, hacemos que todos los trabajos lleguen en el instante más temprano posible teniendo en cuenta su *jitter*. De este modo, garantizamos que la interferencia sobre los trabajos anteriores será más pesimista que la real. Por otro lado incrementamos artificialmente su tiempo de ejecución para garantizar que la influencia sobre los trabajos posteriores será también peor que la real. Demostraremos rigurosamente que el sistema S' sirve para extraer conclusiones sobre el sistema S, desde el lado del pesimismo.

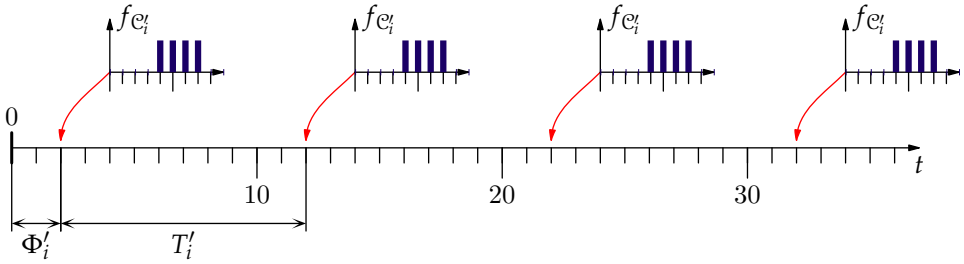
A modo de ejemplo, supongamos que tenemos una tarea τ_i cuyos parámetros son $\Phi_i = 3$, $T_i = 10$, $\mathcal{C}_i = U[2,5]$, $\mathcal{J}_i = [-1,1]$. Su plazo no es relevante. La expresión del *jitter* $[-1,1]$, indica que la tarea podría llegar con un *jitter* de -1, 0 ó 1, con probabilidades desconocidas. La figura 5.12(a) muestra esto de forma gráfica. Los rectángulos grises son los intervalos de tiempo en los cuales el trabajo podría llegar.

La transformación propuesta convertiría a esta tarea en otra, de parámetros: $\Phi_i = 2$, $T_i = 10$, $\mathcal{C}_i = U[4,7]$. El periodo es el mismo que el original. La fase es la original más el *jitter* mínimo, que es -1. El tiempo de ejecución de la tarea

transformada es igual al original más la diferencia entre el *jitter* máximo y el mínimo, que es 2. La nueva distribución del tiempo de ejecución es igual a la original, simplemente desplazada dos unidades a la derecha. La figura 5.12(b) muestra gráficamente las llegadas de esta tarea transformada.



(a) Llegadas de trabajos con *jitter* en el sistema original



(b) Llegadas deterministas en el sistema transformado

Figura 5.12.: Transformación de un sistema con *jitter* aleatorio en otro con llegadas deterministas

Para demostrar que el análisis de este sistema es más pesimista que el del original, procederemos en varias fases. Primero demostraremos que si en una secuencia dada de trabajos, con instantes de llegada deterministas, movemos el instante de llegada de uno de ellos a la izquierda, a la vez que incrementamos su tiempo de ejecución en la cantidad que lo hemos movido, la nueva secuencia de trabajos resultante tiene peores tiempos de respuesta en todos sus trabajos. A partir de este resultado, demostraremos que en un sistema con *jitter*, si cambiamos uno solo de sus trabajos aplicándole la transformación propuesta, el análisis resultante es más pesimista. Finalmente, repitiendo este resultado para cada trabajo del sistema se llega a la conclusión deseada.

Teorema 10. *Sea la secuencia de trabajos* $\Gamma_1, \Gamma_2, \ldots$, *con tiempos de ejecución* $\mathcal{C}_1, \mathcal{C}_2, \ldots$ *y cuyos instantes de llegada deterministas son* $\lambda_1, \lambda_2, \ldots$. *Si se adelanta la llegada de uno de los trabajos,* Γ_j *al instante* $(\lambda_j - \Delta)$, *con* $\Delta > 0$, *a la vez que se incrementa el tiempo de ejecución de dicho trabajo en* Δ *unidades, la nueva secuencia*

de trabajos tiene peores tiempos de respuesta (en el sentido estocástico dado en la definición de "peor").

Demostración. La prueba se divide en tres partes. En primer lugar se demostrará que el tiempo de respuesta empeora para los trabajos con índice menor que j. En segundo lugar, se demostrará que también empeora para los que tienen índice mayor que j. Finalmente, se demuestra que también empeora para el propio trabajo Γ_j que ha sufrido el desplazamiento.

Parte 1. La figura 5.13 muestra la reordenación de los trabajos como consecuencia del desplazamiento sufrido por Γ_j. Al desplazar λ_j hacia la izquierda Δ unidades, hay una serie de trabajos que originalmente eran anteriores a Γ_j (inclusive), pero que ahora son posteriores a él. Llamamos Γ_{j-k} al primero de estos trabajos (ver figura). Se asume que en el intervalo entre $(\lambda_j - \Delta)$ y λ_j no llegan trabajos con prioridad igual a la del trabajo Γ_j. Esta asunción es cierta siempre que el *jitter* sea inferior al periodo de las tareas, para el caso de asignaciones de prioridad RM, o a sus plazos, para el caso EDF, lo que parece razonable.

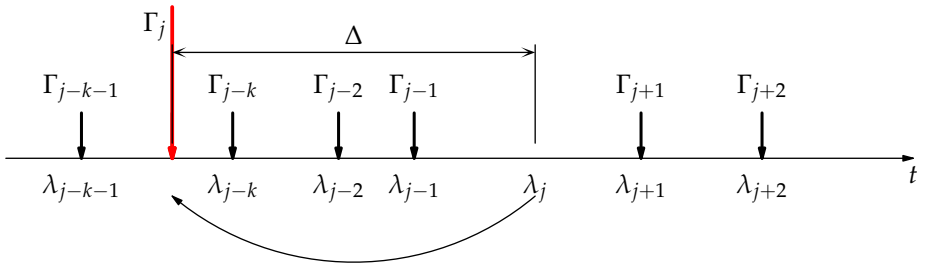


Figura 5.13.: Reordenación de los trabajos al adelantar uno de ellos

Es evidente que para los trabajos anteriores a Γ_{j-k}, el tiempo de respuesta será peor que si Γ_j no se hubiera adelantado, puesto que se aumentan las probabilidades de que Γ_j cause interferencia sobre ellos. Más formalmente, cuando se calcula la PF del tiempo de respuesta de Γ_{j-k-1}, usando el método "partir, convolucionar y juntar", si el trabajo Γ_j ya causaba interferencia en su posición original, la causará también en la nueva posición adelantada, y si no la causaba, ahora cabe la posibilidad de que sí la cause. Por otro lado, el tiempo de ejecución de Γ_j es mayor en la nueva posición, con lo que la interferencia será mayor. Por ambas razones los tiempos de respuesta de los trabajos anteriores a Γ_{j-k-1} serán peores a causa del adelanto de Γ_j.

Para los trabajos entre Γ_{j-k} y Γ_{j-1}, el adelanto de Γ_j tiene dos efectos: de un lado se disminuye la interferencia que estos trabajos sufren (ya que Γ_j no les expulsará), pero de otro lado se incrementa la carga existente cuando estos trabajos entran en ejecución, ya que Γ_j llegó antes que ellos incrementando esta carga. Por

otro lado, ya que el tiempo de ejecución del nuevo Γ_j es como mínimo igual a Δ, se garantiza que la carga añadida en $(\lambda_j - \Delta)$ no se habrá extinguido cuando los trabajos $\Gamma_{j-k}, \ldots, \Gamma_{j-1}$ llegan. De modo que estos trabajos sufrirán con probabilidad 1 la interferencia de Γ_j, si bien en forma de carga inicial, en vez de expulsión. En todo caso esta situación es peor que la original, en la que cabía la posibilidad de que estos trabajos no sufrieran expulsión de Γ_j si terminaban su ejecución antes de que éste llegara.

Parte 2. Los trabajos con índice superior a j, como vemos en la figura 5.13, parece que comienzan con una carga inicial inferior debido al adelanto de Γ_j. Pero esto no es así debido a que el tiempo de ejecución de Γ_j también se ha incrementado en Δ unidades. La carga en el instante λ_j^+ es exactamente la misma que si Γ_j no se hubiera adelantado, ya que las Δ unidades extra de tiempo de ejecución de Γ_j se han consumido durante el lapso de tiempo entre su llegada y λ_j. Así pues, los tiempos de respuesta de los trabajos $\Gamma_{j+1}, \Gamma_{j+2}, \ldots$ no se ven influidos por el adelanto de Γ_j. Pero, por la propiedad reflexiva de la propiedad "peor que", podemos decir que son peores que los originales (aunque en realidad son los mismos).

Parte 3. Veamos ahora qué ocurre con el propio trabajo Γ_j. Como consecuencia de haberse adelantado, la carga que encontrará a su llegada no incluye la contribución de los trabajos $\Gamma_{j-k}, \ldots, \Gamma_{j-1}$. Sin embargo todos estos trabajos le causarán interferencia, ya que al ser el nuevo tiempo de ejecución de Γ_j mayor que Δ, Γ_j estará en ejecución con probabilidad uno cuando $\Gamma_{j-k}, \ldots, \Gamma_{j-1}$ lleguen. Por tanto los tiempos de ejecución de estos trabajos se sumarán al de Γ_j para aumentar su tiempo de respuesta. El tiempo de respuesta será como mínimo igual a la carga existente en $(\lambda_j - \Delta)$ más el propio tiempo de computación de Γ_j, más el de todos los trabajos $\Gamma_{j-k}, \ldots, \Gamma_{j-1}$. Podríamos por tanto decir que:

$$\mathcal{R}_j^{\langle 0 \rangle \prime} = \mathcal{W}(\lambda_j - \Delta) + \sum_{i=1}^{k} \mathcal{C}_{j-i} + \mathcal{C}'_j$$

y a partir de este $\mathcal{R}_j^{\langle 0 \rangle \prime}$ aplicar el método "partir, convolucionar y juntar", siendo el primer trabajo a considerar el que llega en λ_{j+1}. Demostraremos a continuación que el trabajo Γ'_j sufrirá las mismas interferencias que las que habría sufrido de no haber llegado adelantado, y que su tiempo de respuesta es peor.

Llamemos, por brevedad, $\mathcal{X}$ a la variable aleatoria:

$$\mathcal{X} = \mathcal{W}(\lambda_j - \Delta) + \sum_{i=1}^{k} \mathcal{C}_{j-i}$$

Tenemos por tanto que: $\mathcal{R}_j^{\langle 0\rangle\prime} = \mathcal{X} + \mathcal{C}'_j$. Por otro lado $\mathcal{C}'_j = \mathcal{C}_j + \Delta$, por lo que finalmente:

$$\mathcal{R}_j^{\langle 0\rangle\prime} = \mathcal{X} + \Delta + \mathcal{C}_j \tag{5.4}$$

y ya que Δ es una cantidad fija, el efecto de sumarlo a una variable aleatoria es desplazar hacia la derecha el PF de ésta. Así que para obtener $\mathcal{R}_j^{\langle 0\rangle\prime}$ podríamos convolucionar la PF de $\mathcal{X}$ con la de $\mathcal{C}_j$ y desplazar el resultado Δ unidades a la derecha.

Por otro lado, para calcular el $\mathcal{R}_j^{\langle 0\rangle}$ del trabajo original, si no se hubiera adelantado, sería necesario sumar la carga existente en λ_j con el tiempo de ejecución del propio trabajo. Es decir:

$$\mathcal{R}_j^{\langle 0\rangle} = \mathcal{W}(\lambda_j) + \mathcal{C}_j \tag{5.5}$$

Comparando la eq. (5.5) con la eq. (5.4) vemos que son bastante parecidas, y se puede demostrar que $\mathcal{R}_j^{\langle 0\rangle\prime} \succcurlyeq \mathcal{R}_j^{\langle 0\rangle}$, debido a dos razones:

1. De un lado, $\mathcal{X} \succcurlyeq \mathcal{W}(\lambda_j)$, ya que $\mathcal{X}$ es la carga que había en $(\lambda_j - \Delta)$, más las cargas de los trabajos intermedios $\Gamma_{j-k}, \ldots, \Gamma_{j-1}$, mientras que $\mathcal{W}(\lambda_j)$ es esto mismo menos el tiempo transcurrido Δ.
2. Además, $\mathcal{R}_j^{\langle 0\rangle\prime}$ incorpora el sumando $+\Delta$ que hace que $\mathcal{X}$ se desplace Δ unidades a la derecha, lo que la hace definitivamente peor.

La figura 5.14 muestra la función de probabilidad acumulada de $\mathcal{R}_j^{\langle 0\rangle}$ comparada a la de $\mathcal{R}_j^{\langle 0\rangle\prime}$. Se muestra también la $\mathcal{R}_j^{\langle 0\rangle\prime}$ antes de desplazarse Δ unidades a la derecha, para remarcar que incluso sin este desplazamiento ya era peor debido a la razón 1. Las acotaciones sobre la figura son para un razonamiento posterior.

Veamos ahora que además el trabajo adelantado Γ'_j sufrirá las mismas interferencias que las que sufriría de no haberse adelantado. Es decir, si existía una probabilidad no nula de que el trabajo Γ_j fuese interferido por Γ_{j+1}, también habrá una probabilidad no nula de que Γ'_j sea interferido por Γ_{j+1}, y así sucesivamente para cualquier trabajo futuro.

Si el trabajo Γ_j puede ser interferido por Γ_{j+1}, esto quiere decir que su tiempo de respuesta puede llegar a ser mayor que $(\lambda_{j+1} - \lambda_j)$, es decir, se cumple

$$\mathbb{P}\{\mathcal{R}_j^{\langle 0\rangle} > (\lambda_{j+1} - \lambda_j)\} > 0$$

Queremos demostrar que si esto es así, entonces también Γ'_j podrá sufrir esta interferencia, esto es, su tiempo de respuesta puede ser mayor que $(\lambda_{j+1} - \lambda_j - \Delta)$, es decir, que se deberá cumplir:

$$\mathbb{P}\{\mathcal{R}_j^{\langle 0\rangle\prime} > (\lambda_{j+1} - \lambda_j + \Delta)\} > 0$$

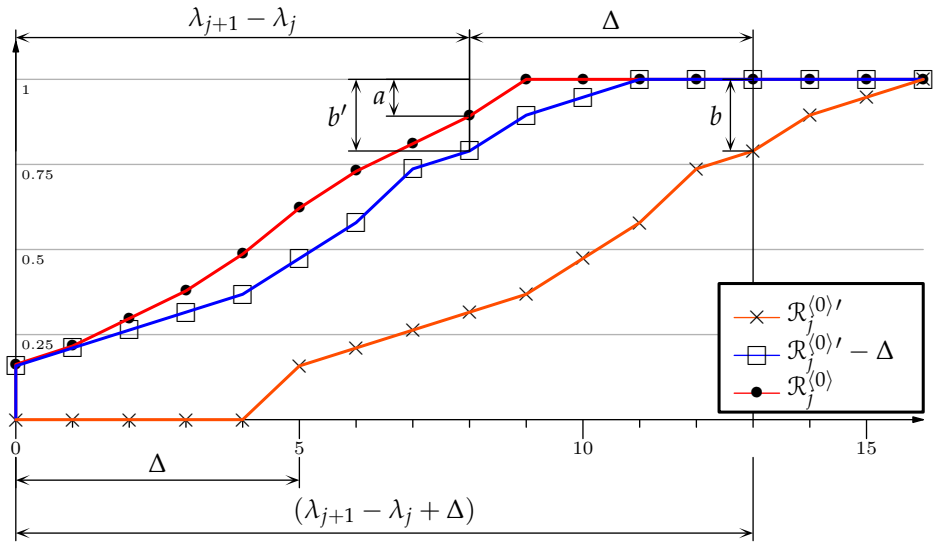


Figura 5.14.: Gráfico de apoyo a la demostración del teorema 10

Esto es cierto como se deduce fácilmente de la figura 5.14. Supongamos que Δ es 5, y que la distancia entre λ_j y λ_{j+1} es 8. Buscamos pues la probabilidad de que $\mathcal{R}_j^{\langle 0\rangle\prime}$ sea mayor que 13. Esta probabilidad es la cantidad marcada como b en la figura. y es la misma que b', debido al desplazamiento de Δ unidades. Por otro lado, b' ha de ser mayor que a debido a que, como ya demostramos, $\mathcal{X}$ era peor que $\mathcal{W}(\lambda_j)$. Pero a es la probabilidad de que $\mathcal{R}_j^{\langle 0\rangle}$ sea mayor que $\lambda_{j+1} - \lambda_j$. De donde se deduce que si esta probabilidad es no nula, $a > 0$, y entonces $b > 0$, como queríamos demostrar.

De lo anterior se desprende que para calcular $\mathcal{R}'_j$ son necesarias exactamente las mismas operaciones "partir, convolucionar y juntar" que para calcular $\mathcal{R}_j$, pero ya que se parte de una inicial peor, al final $\mathcal{R}'_j$ será también peor. □

El teorema anterior muestra por tanto que si adelantamos la llegada de un trabajo, pero a la vez incrementamos artificialmente su tiempo de ejecución en una cantidad igual a lo que habíamos adelantado la llegada, el nuevo sistema presenta un análisis más pesimista que el original. Usaremos este resultado en relación con el *jitter* en la proposición siguiente.

Proposición 7. *Supongamos una secuencia de trabajos $\Gamma_1, \Gamma_2, \ldots$ todos ellos con instantes de llegada deterministas $\lambda_1, \lambda_2, \ldots$ excepto uno de ellos, Γ_j, cuyo instante de llegada presenta* jitter. *Para este trabajo, el instante real de llegada, x, será aleatorio alrededor de un instante teórico λ_j, es decir, $x = \lambda_j + \mathcal{J}_j$.*

Si sustituimos el trabajo Γ_j por otro Γ'_j con instante de llegada determinista: $\lambda'_j = \lambda_j + J_j^{min}$ y con tiempo de ejecución $\mathcal{C}'_j = \mathcal{C}_j + (J_j^{max} - J_j^{min})$, entonces el nuevo sistema tiene peores tiempos de respuesta que el original para todos sus trabajos.

Demostración. El verdadero tiempo de respuesta del sistema original podría calcularse, al menos desde un punto de vista teórico, mediante la técnica exhaustiva de analizar cada posible instancia del sistema. Es decir:

- Para cada posible instante de llegada, x, del trabajo Γ_j
  - Plantear una instancia del problema, que denotaremos por S_x, en la que Γ_j llega en ese instante. La probabilidad de que esta instancia aparezca viene dada por la PF del *jitter*. Llamemos p_x a ésta probabilidad, incluso si es desconocida.
  - Resolver el sistema S_x (que tiene ya llegadas deterministas) y obtener así las PF de los tiempos de respuesta de todos los trabajos. Llamemos $\mathcal{R}_{k,x}$ al tiempo de respuesta del trabajo Γ_k obtenido al resolver el la instancia del sistema S_x.
- La solución al sistema S es un promedio ponderado de las soluciones a todas sus instancias S_x, siendo el factor de ponderación la probabilidad de cada uno de estos sistemas. Es decir, la función de probabilidad de los tiempos de respuesta se obtendría mediante la fórmula:

$$f_{\mathcal{R}_k}(\cdot) = \sum_x p_x f_{\mathcal{R}_{k,x}}(\cdot) \qquad \forall k \tag{5.6}$$

Si comparamos el sistema S' del enunciado del teorema con cada uno de los sistemas S_x, vemos que S' es más pesimista que cualquiera de ellos, pues se ha adelantado la llegada de Γ_j y se ha incrementado su tiempo de ejecución en una cantidad igual o mayor al adelanto sufrido. En virtud del teorema 10 los tiempos de respuesta de todos los trabajos de S' serán peores que los de cualquier sistema S_x. Por tanto:

$$\mathcal{R}'_k \succcurlyeq \mathcal{R}_{k,x} \qquad \forall k, x$$

Si $\mathcal{R}'_k$ es peor que cualquier $\mathcal{R}_{k,x}$, también será peor que su ponderación (propiedad demostrada en la proposición 6, pág. 139), de donde $\mathcal{R}'_k \succcurlyeq \mathcal{R}_k$ para cualquier k, que es lo que se quería demostrar. □

La técnica demostrativa de la proposición anterior puede aplicarse para demostrar el siguiente teorema, de mayor interés práctico:

Teorema 11. *Sea una secuencia de trabajos $\Gamma_1, \Gamma_2, \ldots$ con unos instantes de llegada teóricos $\lambda_1, \lambda_2, \ldots$ y unos instantes de llegada reales $t_j = \lambda_j + \mathcal{J}_j$, siendo $\mathcal{J}_j$ una variable aleatoria.*

Si sustituimos uno solo de los trabajos, Γ_k, por otro Γ'_k con instante de llegada determinista: $\lambda'_k = \lambda_k + J_k^{min}$ y con tiempo de ejecución $\mathcal{C}'_k = \mathcal{C}_k + (J_k^{max} - J_k^{min})$, entonces el nuevo sistema tiene peores tiempos de respuesta que el original para todos sus trabajos.

Demostración. La demostración usa una técnica similar a la de la proposición anterior. Para calcular la distribución "exacta" de un tiempo de respuesta es necesario plantear todas las posibles instancias del sistema, es decir, todas las combinaciones posibles de llegadas de todos los trabajos. El número de instancias a analizar crecería factorialmente con el número de trabajos, pero en todo caso sería un número finito, por lo que desde el punto de vista teórico es posible realizar tal cálculo.

Imaginemos una "instancia" del sistema, en la que tenemos fijados los instantes de llegada de todos los trabajos, excepto el de Γ_k, al cual dejamos con un instante de llegada "borroso" (es decir, no instanciamos este trabajo). Tenemos entonces un sistema como el del enunciado de la proposición anterior. En ella se demostró que cambiar Γ_k por Γ'_k, lleva a un caso peor que esa instancia del problema.

Puesto que lo anterior es cierto para cada posible instancia del problema, se concluye que al cambiar Γ_k por Γ'_k se obtiene un problema que es peor que el original en cada una de sus posibles instancias. Por tanto la solución final, que sería una ponderación de estas soluciones, será también peor. □

Corolario 4. *Dado un sistema S compuesto por N tareas periódicas, cada una de ellas con un* jitter *aleatorio $\mathcal{J}_i$, si modificamos el sistema usando el conjunto de ecuaciones (5.2), el nuevo sistema S' resultante tiene peores tiempos de respuesta (en sentido estadístico) en todos sus trabajos.*

Demostración. Basta aplicar el teorema anterior a cada uno de los trabajos. Cada vez que un trabajo con *jitter* se cambia por otro con llegada determinista y su tiempo de ejecución aumentado, el nuevo análisis es más pesimista. Si se cambian todos los trabajos, el resultado será más pesimista. □

Este corolario es el que nos permite resolver sistemas con *jitter* usando la técnica descrita, y con garantías de que la solución obtenida es más pesimista que la del sistema original.

5.4. Tareas aperiódicas y esporádicas

El modelo inicial suponía que los instantes de activación de cada tarea eran completamente deterministas. En el apartado anterior hemos ampliado el modelo para que tenga en cuenta una pequeña variación en el instante de activación, a costa de introducir pesimismo en el análisis. Cuanto mayor sea la posible variación del instante de activación (*jitter*), mayor será el pesimismo introducido.

Cuando los instantes de activación son completamente aleatorios, como es el caso de las tareas aperiódicas, el enfoque anterior deja de ser útil. No es posible en este caso modelar el instante de llegada como un "instante teórico más un

jitter", a menos que estemos dispuestos a utilizar valores de *jitter* arbitrariamente grandes, los cuales introducirían un pesimismo inaceptable en el análisis.

El enfoque clásico para tratar con este tipo de tareas consiste en acotar el intervalo mínimo entre llegadas (tareas esporádicas), y asumir un "peor caso" en el que todas las tareas llegan con esa separación. Es decir, se las trata como periódicas siendo el periodo igual al tiempo mínimo entre llegadas. Este método también introduce un gran pesimismo, y no es aplicable a las tareas aperiódicas en las cuales el tiempo entre llegadas no tiene cota inferior.

Hemos visto en la sección sobre trabajo relacionado (pág. 21) algunos enfoques estocásticos al problema. En particular, el enfoque de Abeni y Buttazzo [2001] es prometedor, y además puede ser fácilmente integrado con el análisis propuesto en esta tesis.

En el enfoque de Abeni y Buttazzo [2001] es necesario modificar el planificador, de modo que garantice un tiempo de servicio cada cierto periodo de tiempo para cada tarea. De este modo se separan los parámetros de planificación de las tareas de sus parámetros intrínsecos (instantes de llegada, tiempo de ejecución). Con independencia de cuál sea el tiempo de ejecución de las tareas y de cuáles sean sus instantes de llegada, el planificador asegurará que la tarea τ_i podrá ejecutarse durante Q_i^s unidades de tiempo cada T_i^s unidades de tiempo, siendo estos dos parámetros decididos por el diseñador del sistema. Un planificador así es el "Servidor de ancho de banda constante" [Abeni y Buttazzo, 1998]. Una vez que el planificador ha garantizado esto, las tareas aperiódicas quedan "aisladas" unas de otras desde el punto de vista del análisis.

En nuestro caso, sería posible tener un conjunto de tareas periódicas junto con una serie de tareas aperiódicas o esporádicas. Las tareas aperiódicas serían servidas por este servidor de ancho de banda constante. En realidad habría tantos servidores como tareas aperiódicas. Cada uno de estos servidores sería tratado por nuestro análisis como una tarea periódica, de periodo T_i^s y de tiempo de ejecución determinista Q_i^s, de cara a analizar su influencia sobre las tareas periódicas. De este modo, utilizando el análisis ya explicado en esta tesis, se obtendrían las probabilidades de pérdida de plazos para las tareas periódicas del sistema.

En cuanto a las tareas aperiódicas, cada una de ellas es analizable de forma independiente al resto del sistema (pues el servidor asegura su aislamiento), y para obtener sus probabilidades de pérdidas de plazo se recurriría al modelo de colas expuesto en el trabajo de [Abeni y Buttazzo, 2001].

Esta idea aún puede elaborarse más, puesto que el tiempo de ejecución del servidor no es en realidad determinista. El diseñador fija el valor de Q_i^s, que es el tiempo *máximo* que se ejecutará el servidor de la tarea aperiódica τ_i en cada ciclo de servicio (ciclos que se repiten cada T_i^s unidades). Si las instancias de la tarea que llegan en ese ciclo requieren menos tiempo, el servidor puede finalizar antes. De modo que el análisis se podría refinar tratando de obtener la distribución estadística de Q_i^s, de cara a reducir el pesimismo en el cálculo de la interferencia

que este servidor causa sobre las restantes tareas periódicas. Esta es una línea de investigación abierta.

5.5. Dependencia estadística entre tareas

El modelo ha supuesto que los tiempos de ejecución de cada trabajo son variables aleatorias independientes de los tiempos de ejecución de los restantes trabajos. En la práctica esto puede no ser así, ya que pueden aparecer *correlaciones*. Por ejemplo, podría ocurrir que, cuando una instancia de una tarea requiera un tiempo de ejecución alto, haya mayores probabilidades de que la siguiente instancia también requiera un tiempo alto. O la correlación podría ser inversa.

Al ignorar la posible correlación, si en realidad existía una, las probabilidades que obtendremos para las pérdidas de plazo no serán correctas. Y lo peor es que no serían necesariamente pesimistas ni tampoco optimistas, como mostraremos a continuación con un ejemplo.

El problema de la correlación aparece al calcular la función de probabilidad de la suma de dos variables aleatorias $\mathcal{X}$ e $\mathcal{Y}$. En el análisis desarrollado en esta tesis, la PF de la suma se ha obtenido mediante la convolución de las PFs de $\mathcal{X}$ e $\mathcal{Y}$, pero esta operación sólo es válida si las variables $\mathcal{X}$ e $\mathcal{Y}$ son estadísticamente independientes.

En el caso más general posible, la probabilidad de que la suma $\mathcal{X}+\mathcal{Y}$ tome el valor z, se calcula simplemente sumando las probabilidades de todas las combinaciones de $\mathcal{X}$ e $\mathcal{Y}$ que al sumarse produzcan z, es decir:

$$\mathbb{P}\{\mathcal{X}+\mathcal{Y}=z\} = \sum_{i=-\infty}^{\infty} \mathbb{P}\{\mathcal{X}=i \wedge \mathcal{Y}=z-i\} \tag{5.7}$$

En el caso particular en que las variables $\mathcal{X}$ e $\mathcal{Y}$ sean estadísticamente independientes, se tendrá que

$$\mathbb{P}\{\mathcal{X}=i \wedge \mathcal{Y}=z-i\} = \mathbb{P}\{\mathcal{X}=i\} \cdot \mathbb{P}\{\mathcal{Y}=z-i\}$$

y en este caso la ecuación (5.7) se convierte en una convolución.

Sin embargo, si las variables no son estadísticamente independientes, la probabilidad de la intersección $(\mathcal{X}=i \wedge \mathcal{Y}=z-i)$ no puede obtenerse a partir de las probabilidades de $(\mathcal{X}=i)$ y $(\mathcal{Y}=z-i)$, sino que debe ser suministrada como dato. Naturalmente esto plantea el problema de cómo obtener dicho dato. Sobre todo si se tiene en cuenta que en muchos casos las variables aleatorias que sumamos no son dos tiempos de ejecución de dos tareas, sino la carga en un instante con el tiempo de ejecución de un trabajo. ¿Cómo obtener la correlación entre estas dos variables aleatorias? La respuesta no parece trivial, y no se ha abordado en esta tesis.

Supongamos, de todas formas, que el problema ha sido resuelto, y que disponemos de las probabilidades de cada posible par $(\mathcal{X}, \mathcal{Y})$. Esta información vendría representada por una función de dos variables: la función de probabilidad conjunta de $\mathcal{X}$ e $\mathcal{Y}$

$$f_{\mathcal{X},\mathcal{Y}}(x,y) = \mathbb{P}\{\mathcal{X}=x \wedge \mathcal{Y}=y\}$$

por lo que la PF de la suma podría calcularse a partir de esta distribución conjunta

$$f_{\mathcal{X}+\mathcal{Y}}(z) = \sum_{i=-\infty}^{\infty} f_{\mathcal{X},\mathcal{Y}}(i, z-i) \tag{5.8}$$

En nuestro problema particular, ya que las variables aleatorias son siempre positivas y acotadas por un valor máximo, la función de probabilidad conjunta podría almacenarse en un *array* bidimensional. Mostramos un ejemplo en la figura 5.15. La suma por filas (o por columnas) nos produce la "función de probabilidad marginal" de la variable $\mathcal{X}$ (o $\mathcal{Y}$), es decir, las probabilidades de que esta variable tome cualquier valor dado, con independencia de la otra variable.

$\mathcal{X}$ \ $\mathcal{Y}$	0	1	2	3	4	$f_{\mathcal{X}}$
0	0	0.06	0.02	0.08	0.04	0.2
1	0.04	0.02	0.1	0	0.04	0.2
2	0.02	0.08	0.02	0.08	0	0.2
3	0.02	0	0.06	0	0.12	0.2
4	0.02	0.04	0.1	0.04	0	0.2
$f_{\mathcal{Y}}$	0.1	0.2	0.3	0.2	0.2	1

Figura 5.15.: Ejemplo de función de probabilidad conjunta, y probabilidades marginales

La función de probabilidad de la suma $\mathcal{X} + \mathcal{Y}$ se obtendría aplicando la ecuación (5.8), que equivale a sumar los elementos del *array* a lo largo de las diagonales NE–SO, como se muestra en la figura 5.16.

En el modelo utilizado en esta tesis, tan sólo conocemos las distribuciones marginales, pero no la función de probabilidad conjunta. Es decir, sabemos lo que debe sumar cada fila y cada columna de la tabla, pero no los valores particulares de cada elemento. Al asumir independencia entre variables, estamos rellenando cada elemento de la tabla con el producto de las correspondientes probabilidades marginales, como se muestra en la figura 5.17. Sumar las diagonales en esta segunda figura equivale a realizar la convolución.

El problema es que las probabilidades obtenidas por convolución no son "peores" (en el sentido estadístico definido en esta tesis) que las obtenidas a partir de la "verdadera" función de probabilidad conjunta. La figura 5.18 compara las CDF

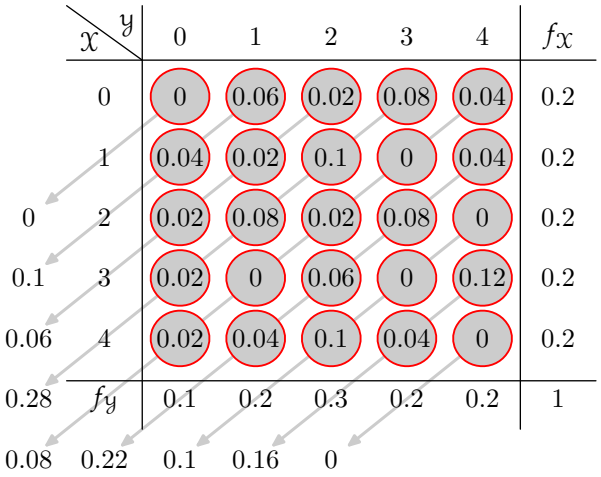


Figura 5.16.: Ejemplo del cálculo de la probabilidad de la suma, a partir de la función de probabilidad conjunta

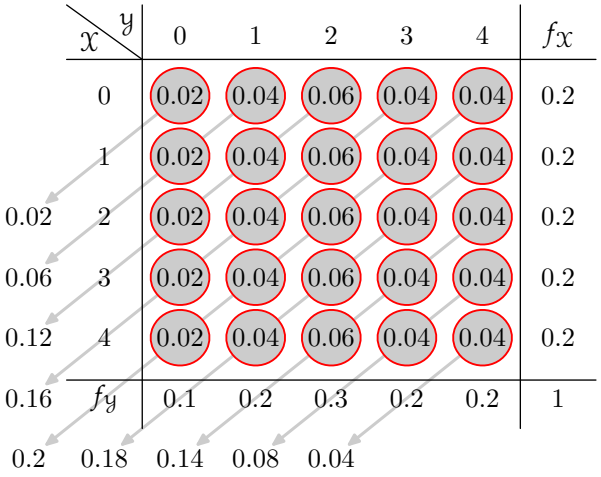


Figura 5.17.: Ejemplo del cálculo de la probabilidad de la suma, suponiendo independencia estadística

de ambos resultados y se observa que se cruzan, por lo que ninguna es peor que la otra.

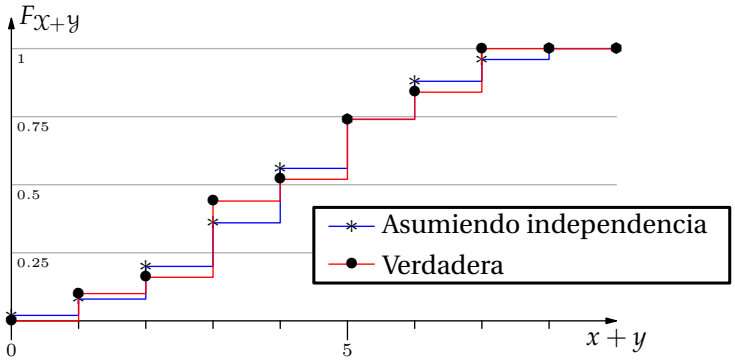


Figura 5.18.: Comparación entre considerar la correlación o asumir independencia

Se puede pensar en ampliar el modelo para incluya más información estadística. Los tiempos de ejecución de las tareas ya no vendrían dados por una única función de probabilidad, sino por una distribución conjunta multidimensional. Este enfoque tiene dos importantes desventajas prácticas:

- El coste computacional del análisis aumenta aún más, y posiblemente de forma exponencial, ya que es necesario introducir funciones de distribución multidimensionales, y el número de dimensiones podría llegar a ser en última instancia igual al número de trabajos por hiperperiodo[9]
- Queda por resolver cómo se obtendría toda esta información estadística para alimentar el modelo. Si ya el encontrar la PF de un tiempo de ejecución para una única tarea parece un problema suficientemente complejo [Bernat *et al.*, 2002], el tratar de hallar las funciones de distribución conjuntas de varias tareas puede ser imposible.

Así pues, parece poco práctico tratar de conocer las funciones de probabilidad conjuntas de todas las variables aleatorias que participan en el modelo y en su análisis. En cambio, las probabilidades marginales son perfectamente conocidas. Cabe plantearse la pregunta de si sería posible "inventar" una función de probabilidad conjunta, a partir de las probabilidades marginales, que si bien no coincida con la auténtica, que es desconocida, al menos sea "peor" que ella. Es decir, que la PF de la suma utilizando esta función conjunta inventada salga "peor" (en el sentido definido en esta tesis) que la que se obtendría con la auténtica, sea cual sea ésta.

En el contexto de la obtención estadística del *WCET*, Bernat *et al.* se han encontrado con el mismo problema. Los citados autores han inventado un nuevo

[9] Asumiendo que no hay correlación estadística de un hiperperiodo a otro

operador de convolución (que denominan "convolución sesgada"[10]) que no necesita conocer la función de probabilidad conjunta, sino tan solo las marginales. El método se basa en la idea de que, de entre todas las posibles funciones de probabilidad conjuntas que son coherentes con las funciones marginales dadas, habrá una que sea "la peor" (en un sentido claramente definido por los autores). Se demuestra que tal función existe y se proporciona un algoritmo para obtenerla.

El problema radica en que la definición de "peor" suministrada en dicho artículo no coincide con la de esta tesis. Para Bernat *et al.* una distribución es peor que otra si su centro de gravedad está más a la derecha. Formalmente, definen la función f "peor" como aquella que maximice el valor de $\beta(f)$ definida como:

$$\beta(f) = \sum_{i=-\infty}^{\infty} i^2 f(t)$$

Sin embargo esta definición no garantiza que la CDF de la función "peor" esté siempre por debajo de las otras, como exigimos en esta tesis. De hecho, se cruza con ellas, como mostraremos a continuación con un ejemplo. Aplicando la técnica de la convolución sesgada a las funciones marginales de la figura 5.16, obtenemos una función conjunta "inventada" que es coherente con dichas distribuciones marginales, y que se muestra en la figura 5.19. No mostramos aquí los detalles de cómo se ha encontrado este resultado, basta seguir el algoritmo claramente detallado en [Bernat *et al.*, 2002]. Una vez obtenida esta función conjunta artificial, que el lector puede verificar es coherente con las distribuciones marginales dadas, la probabilidad de la suma se obtiene como en los demás casos sumando las diagonales.

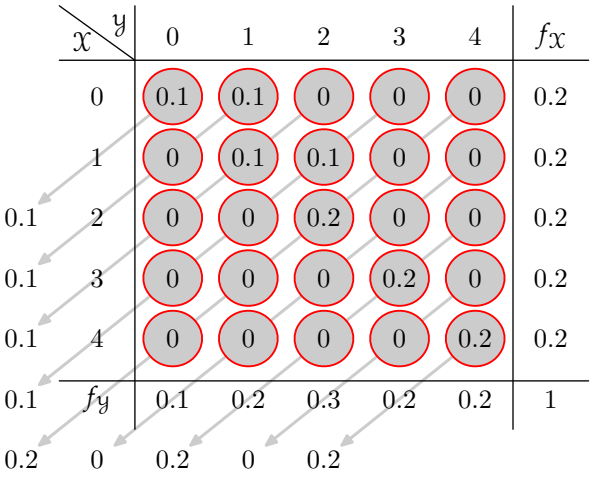
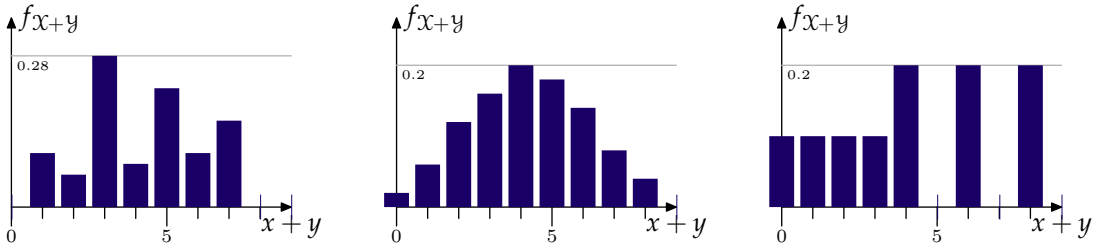


Figura 5.19.: Cálculo de la probabilidad de la suma usando el operador "convolución sesgada"

El resultado obtenido tiene la máxima $\beta(f)$ entre todas las funciones conjuntas f compatibles con las funciones marginales, tal como se demuestra en [Bernat

[10] *biased*

et al., 2002]. La figura 5.20 compara las PF obtenidas por los tres métodos mostrados hasta ahora, esto es, usando la "verdadera" función conjunta, asumiendo independencia estadística, o utilizando la convolución sesgada. Vemos que ésta última es la que produce una PF más "desviada" hacia la derecha, y el lector puede comprobar que es la que tiene un mayor valor de $\beta(f)$.



(a) Usando la verdadera probabilidad conjunta, $\beta(f) = 21{,}08$

(b) Asumiendo independencia estadística, $\beta(f) = 21{,}2$

(c) Usando la "convolución sesgada", $\beta(f) = 24{,}06$

Figura 5.20.: Comparación de las funciones de probabilidad obtenidas por tres métodos diferentes

Por desgracia, si comparamos las CDF en lugar de las PF, vemos que la obtenida por convolución sesgada no es peor que las demás (ver figura 5.21). De hecho, las tres se cruzan entre sí, por lo que no hay ninguna peor que otra. En mi opinión, la definición de "peor" usada en [Bernat *et al.*, 2002] no es adecuada para obtener cotas superiores a la probabilidad de perder plazos, al menos desde un punto de vista teórico. Desde un punto de vista práctico tal vez pueda usarse, si se demuestra que, *al menos para valores por encima del plazo*, la función que maximiza $\beta(f_{X+Y})$ está por debajo de cualquier otra f_{X+Y} (coherente con las distribuciones marginales de X e Y). En todo caso, este es un punto a investigar.

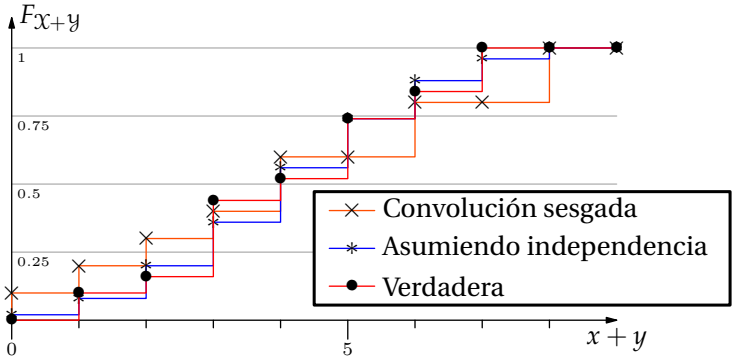


Figura 5.21.: Comparación de las funciones de distribución acumuladas obtenidas por tres métodos diferentes

Siguiendo la misma idea de Bernat *et al.* podríamos preguntarnos si, entre todas las funciones de probabilidad conjunta que son coherentes con las distribuciones

marginales f_X, f_Y, habrá alguna que produzca una f_{X+Y} "peor" en el sentido definido en esta tesis. Demostraremos a continuación que, por desgracia, tal cosa no existe.

Definición 21. *Diremos que una función de probabilidad conjunta $f_{X,Y}(\cdot,\cdot)$ es peor que otra $f'_{X,Y}(\cdot,\cdot)$, y lo denotaremos por $f_{X,Y}(\cdot,\cdot) \succcurlyeq f'_{X,Y}(\cdot,\cdot)$, si la función de probabilidad acumulada de $X+Y$ obtenida de f está siempre por debajo de la obtenida de f', es decir, si*

$$\sum_{i=-\infty}^{z}\sum_{j=-\infty}^{\infty} f_{X,Y}(j,i-j) \leq \sum_{i=-\infty}^{z}\sum_{j=-\infty}^{\infty} f'_{X,Y}(j,i-j) \qquad \textit{para todo } z$$

Teorema 12. *Sea G el conjunto de todas las funciones de probabilidad conjunta $f_{X,Y}(\cdot,\cdot)$ que son coherentes con las funciones de probabilidad marginal $f_X(\cdot)$ y $f_Y(\cdot)$.*

La función f tal que $f(\cdot,\cdot) \succcurlyeq f_{X,Y}(\cdot,\cdot)$ para todo $f_{X,Y}(\cdot,\cdot) \in G$ no está en G.

Demostración. Podemos representar la función $f_{X,Y}(\cdot,\cdot)$ mediante una matriz rectangular, que denominaremos $\mathbf{M}$. Lamemos a sus elementos $m_{i,j}$, de modo que $m_{i,j} = f_{X,Y}(i,j)$.

Para que una función $f_{X,Y}(\cdot,\cdot)$ pertenezca a G, debe ser coherente con la función marginal $f_X(\cdot)$, es decir, la suma "fila a fila" de la matriz debe ser igual a $f_X(\cdot)$. Esto debe cumplirse en particular, para la fila cero:

$$\sum_{i\geq 0} m_{0,i} = f_X(0)$$

También debe ser coherente con la función maginal f_Y, para lo cual la suma "columna a columna" debe ser igual a f_Y. En particular para la columna cero:

$$\sum_{j\geq 0} m_{j,0} = f_Y(0)$$

De las ecuaciones anteriores podemos despejar el elemento $m_{0,1}$ y el elemento $m_{1,0}$, respectivamente, y obtener:

$$m_{0,1} = f_X(0) - m_{0,0} - \sum_{i>1} m_{0,i} = k - m_{0,0}$$
$$m_{1,0} = f_Y(0) - m_{0,0} - \sum_{j>1} m_{j,0} = l - m_{0,0}$$

donde, para simplificar la expresión, hemos agrupado el primer sumatorio con el sumando $f_X(0)$, llamando al resultado k, y análogamente el segundo sumatorio con $f_Y(0)$, llamando al resultado l.

La probabilidad de que $\mathcal{X} + \mathcal{Y}$ tome el valor cero, es la probabilidad conjunta de que $\mathcal{X}$ tome el valor cero e $\mathcal{Y}$ también. Esto es el valor de $m_{0,0}$.

La probabilidad de que $\mathcal{X} + \mathcal{Y}$ tome el valor 1, es la suma de las probabilidades de que $(\mathcal{X} = 0) \wedge (\mathcal{Y} = 1)$ y $(\mathcal{X} = 1) \wedge (\mathcal{Y} = 0)$, esto es, la suma $m_{0,1} + m_{1,0}$.

Esto nos permite determinar los dos primeros puntos de la función de probabilidad acumulada de la suma, $F_{\mathcal{X}+\mathcal{Y}}(\cdot)$

$$\begin{aligned} F_{\mathcal{X}+\mathcal{Y}}(0) &= \mathbb{P}\{\mathcal{X} + \mathcal{Y} = 0\} = m_{0,0} \\ F_{\mathcal{X}+\mathcal{Y}}(1) &= \mathbb{P}\{\mathcal{X} + \mathcal{Y} = 0\} + \mathbb{P}\{\mathcal{X} + \mathcal{Y} = 1\} \\ &= m_{0,0} + (m_{0,1} + m_{1,0}) \\ &= k + l - m_{0,0} \end{aligned} \tag{5.9}$$

Esto nos da los dos primeros valores de $F_{\mathcal{X}+\mathcal{Y}}$, y sólo con ellos ya podemos comprobar que no hay forma de encontrar un $m_{0,0}$ que nos garantice un caso "peor" en el sentido definido antes. En efecto, para cualquier valor que demos a $m_{0,0}$, si lo disminuimos en una pequeña cantidad, lograremos una nueva $F'_{\mathcal{X}+\mathcal{Y}}$ que está por debajo de $F_{\mathcal{X}+\mathcal{Y}}$ en la abcisa cero, pero en cambio está por encima en la abcisa 1 (ya que al disminuir $m_{0,0}$, aumentamos $F_{\mathcal{X}+\mathcal{Y}}(1)$ como se deduce de las fórmulas anteriores). Y si aumentamos $m_{0,0}$, lograremos que la nueva F' esté por debajo para la abcisa 1, pero entonces estará por encima para la abcisa cero.

Es por tanto imposible encontrar una $F'_{\mathcal{X}+\mathcal{Y}}$ que esté por debajo de $F_{\mathcal{X}+\mathcal{Y}}$ en todas sus abcisas, y que cumpla las restricciones de la ecuación (5.9), que deben ser cumplidas para garantizar que la función pertenece a G. □

Así pues, si planteamos el problema de analizar un sistema que presenta correlaciones estadísticas entre sus variables, pero en ausencia de información, es decir, disponiendo tan sólo de distribuciones marginales y no de las probabilidades conjuntas, tal sistema no puede ser analizado, ni siquiera desde un punto de vista pesimista, como acabamos de demostrar.

Es por tanto un problema abierto el investigar en cómo la información sobre las correlaciones podría ser añadida al análisis. Probablemente la solución más simple sea que el analista identifique las *causas* de la correlación, y en base a estas causas diseñe el escenario en el cual la correlación es la peor posible, a la hora de obtener las funciones de probabilidad de los tiempos de ejecución de las tareas que se introducirán en el modelo. Es decir, el analista no introduce en el modelo las distribuciones marginales de los tiempos de ejecución, sino las distribuciones *condicionadas* al caso en que son peores. De este modo se podría llevar a cabo un análisis que asuma independencia, que produciría resultados conservadores. Por ejemplo, si se encuentra que una causa de correlación estadística son los fallos de cache, el analista debería buscar el perfil de los tiempos de ejecución de las tareas cuando haya fallos de cache, e introducir estos perfiles en el modelo, en lugar de los perfiles obtenidos "promediando" los casos en que hay fallos de cache con los

casos en los que no los hay (que sería los que obtendría en una medición que no tenga en cuenta el estado de la cache).

5.6. Ampliaciones triviales

El análisis ha asumido siempre un modelo de tareas periódicas simple, en el que cada tarea se caracteriza por un periodo constante. Sin embargo no hay nada en el análisis que fuerce esta limitación. Las tareas pueden tener cualquier patrón de activaciones, con tal de que sea determinista y que presente algún tipo de periodicidad. Por otro lado se ha supuesto un planificador con desalojo, por ser el caso más complejo. Sin embargo la solución para el caso sin desalojo se puede obtener trivialmente a partir del análisis propuesto. En esta sección abordamos estos dos aspectos.

- **Patrón de activaciones**. Las tareas pueden tener cualquier patrón de activaciones determinista y periódico. Esto cubre el caso ya analizado en el que cada tarea tiene un periodo constante, pero también el caso de las tareas periódicas por ráfagas (un periodo indica la separación entre ráfagas y otro la separación entre tareas dentro de la ráfaga) u otros patrones más complejos. Por ejemplo, podríamos suponer un tipo de tarea τ_i que presente un intervalo $T_{i,1}$ entre la primera y segunda activación, seguido de otro intervalo $T_{i,2}$ entre la segunda y la tercera, etc. de forma que este patrón de intervalos comience a repetirse en algún momento. La única limitación es que los instantes de llegada deben ser deterministas y presentar algún tipo de periodicidad.

  Los parámetros que modelan los instantes de activación de las tareas intervienen en el análisis únicamente para determinar qué tarea será la próxima en activarse a partir de un instante dado. Es decir, se necesitan para convertir el modelo de tareas en una secuencia de activaciones de trabajos. Una vez conocida esta secuencia, el análisis procederá "a nivel de trabajo", aplicando los métodos ya descritos. Puesto que cada tarea presenta algún tipo de patrón periódico, la secuencia de trabajos también lo presentará. El hiperperiodo será ahora el intervalo de tiempo entre dos repeticiones de esta secuencia de activaciones. Su longitud ya no será tan sencilla de calcular, pero en todo caso podrá obtenerse. La existencia de un hiperperiodo permitirá aplicar el análisis Markoviano también a este tipo de sistemas.

- **No-expulsión**. Si el planificador no permite expulsión, en el momento que una tarea entre en ejecución ya no podrá sufrir interferencia de ninguna otra tarea. Por tanto su tiempo de respuesta sería igual a la carga de mayor prioridad existente en el instante de su activación, más su propio tiempo

de ejecución. Esto es precisamente lo que denominabamos $\mathcal{R}_i^{\langle 0 \rangle}$ (el tiempo de respuesta de una tarea suponiendo que será expulsada por los siguientes "cero" trabajos). Ya hemos dado el método para obtener la distribución de esta cantidad.

Por tanto los sistemas sin expulsión están cubiertos también dentro de nuestro análisis, de una forma trivial.

6. Coste computacional y resultados experimentales

> No amount of experimentation can ever prove me right; a single experiment can prove me wrong.
>
> *(Albert Einstein)*

En este capítulo se cubren dos objetivos diferentes. Por un lado (sección 6.1) se analiza en detalle el coste computacional del análisis estocástico propuesto en el capítulo 4. Las conclusiones son también válidas con ligeras modificaciones para las extensiones propuestas en el capítulo 5. Se obtiene un límite teórico al crecimiento del coste, y se muestran resultados experimentales (sección 6.2) que corroboran este límite teórico.

En una segunda parte (sección 6.3) se comparan los resultados obtenidos mediante el análisis estocástico propuesto en esta tesis con los de otros métodos propuestos en la literatura, y con los obtenidos por simulación. Se observa que los valores obtenidos con nuestro análisis son mucho más próximos a los simulados que usando cualquier otro método, y que se requiere mucho menos tiempo que en la simulación para llegar a resultados mucho más exactos.

6.1. Coste computacional

El análisis propuesto presenta una gran complejidad computacional, debido al elevado número de convoluciones que es necesario realizar. En esta sección encontraremos el límite asintótico en el número de operaciones necesarias, en función de ciertos parámetros del sistema, y veremos que a pesar del elevado coste, éste no es de tipo exponencial, sino polinómico en función del número de trabajos en un hiperperiodo y del tamaño de las funciones de probabilidad que definen el sistema.

Aún creciendo de forma polinómica, el tiempo necesario para resolver un sistema numéricamente con la ayuda de un computador se va haciendo más y más prohibitivo a medida que crece el tamaño del problema. Resulta por tanto de interés conocer los límites prácticos de aplicabilidad de este análisis, esto es ¿estamos hablando del orden de minutos, de horas, de días? ¿Cuál es el tamaño máximo de

un sistema que podríamos resolver en un tiempo "razonable"? Estudiaremos en la sección 6.1 el tiempo requerido por un ordenador personal de gama media para resolver diferentes tipos de sistemas, y daremos gráficas y tablas que permitirán estimar de antemano el tiempo que ser requerirá para resolver un sistema dado.

6.1.1. Modelo para el análisis del coste computacional

Para estudiar el coste computacional, consideraremos al sistema S definido, no por un conjunto de tareas periódicas, sino como una secuencia de n trabajos $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ cuyos instantes de llegada vienen dados por $\lambda_1, \lambda_2, \ldots, \lambda_n$. Cada trabajo tiene un tiempo de ejecución aleatorio $\mathcal{C}_j$, definido por su función de probabilidad. Cada una de estas funciones de probabilidad viene definida por m puntos, que pueden ser almacenados en un vector. Para simplificar el estudio del coste computacional, asumiremos una implementación ineficiente, en la que los m puntos almacenados representan las probabilidades de que la carga $\mathcal{C}_j$ tome los valores $0, 1, 2, \ldots, C_j^{\max}$. En este caso $C_j^{\max} = m - 1$. Una implementación más eficiente almacenaría sólo los m puntos en los que estas probabilidades fueran distintas de cero.

Asumiremos que los trabajos presentan una periodicidad a nivel de hiperperiodo, es decir, en el instante $\lambda_j + T$ llega una nueva instancia de Γ_j, con la misma distribución de su tiempo de ejecución.

Cualquier modelo de tareas periódicas puede convertirse a un modelo como el anterior, por lo que lo que sigue es válido para el modelo clásico de tareas periódicas.

Para dotar de mayor generalidad al estudio de la complejidad, asumiremos que los instantes λ_j de llegada de los trabajos no siguen ningún patrón regular dentro del hiperperiodo, sino que llegan en instantes aleatorios cuyos intervalos entre llegadas siguen una cierta distribución, que no nos concierne. Sabemos que esto no es así, ya que normalmente los trabajos provienen de tareas periódicas y por tanto sí que hay regularidad en sus llegadas. Sin embargo, para no introducir en el estudio los periodos de las tareas, y cubrir así un sistema que sea de alguna forma el "caso promedio" de todos los sistemas posibles, asumiremos que los intervalos entre llegadas son aleatorios. Llamaremos $\bar{T}$ al tiempo promedio entre llegadas. En sistemas de tareas periódicas, este parámetro se calcularía dividiendo la longitud del hiperperiodo entre el número de trabajos que llegan en él, es decir:

$$\bar{T} = \frac{T}{\sum_{i=0}^{N}(T/T_i)}$$

En nuestro caso de trabajos no periódicos, sería simplemente $\bar{T} = T/n$.

6.1.2. Análisis del coste

Para encontrar el tiempo de respuesta de un trabajo, Γ_j, son necesarios los siguientes pasos:

- Obtener la distribución estacionaria de la carga al principio del hiperperiodo, ya sea mediante análisis Markoviano o mediante aproximación iterativa.
- Avanzar, usando el método "convolucionar y encoger" hasta el instante λ_j en que llega el trabajo bajo consideración.
- Aplicar el método "partir, convolucionar y juntar" para cada uno de los futuros trabajos que pueden causar interferencia al trabajo actual, y que llegan en la ventana $[\lambda_j, \lambda_j + D_j]$. Recordar que no es necesario tomar en consideración los que llegan después de $\lambda_j + D_j$ porque éstos ya no afectan a la probabilidad de perder el plazo.

Para situarnos en el peor caso, supongamos que el trabajo Γ_j del cual queremos obtener su tiempo de respuesta, es el de menor prioridad del sistema. En este caso, todos los trabajos anteriores del hiperperiodo contribuyen a su retraso inicial, y todos los posteriores podrían contribuir a su interferencia, dentro de la ventana $[\lambda_j, \lambda_j + D_j]$.

Con estas premisas, calculemos el número de operaciones que será necesario realizar para obtener la PF del tiempo de respuesta del trabajo Γ_j. Nos centraremos únicamente en el número de operaciones necesarias para realizar las convoluciones, ya que éstas representan el grueso de la carga computacional. Dividiremos el estudio en las tres fases del algoritmo, pero comenzaremos suponiendo que la carga estacionaria al principio del hiperperiodo ya ha sido obtenida de algún modo, y dejaremos para más adelante el estudio del coste de obtener dicha carga estacionaria.

Coste computacional del método "convolucionar y encoger"

Supongamos que la carga al comienzo del hiperperiodo ya ha sido obtenida, y está representada por una función de probabilidad definida por b puntos, almacenada en un vector de modo que el elemento con índice 0 indica la probabilidad de que la carga inicial sea cero, y el elemento con índice $b-1$ (el último) indica la probabilidad de que la carga tome su valor máximo $b-1$. Para calcular a partir de ésta la carga en el instante λ_j, será necesario aplicar j veces el método "convolucionar y encoger", ya que partimos de la hipótesis de que Γ_j tiene menor prioridad que todos los que le preceden.

Cada vez que se realiza una operación "convolucionar", se requieren $2m_1m_2$ operaciones (sumas más restas), siendo m_1 y m_2 el número de puntos de cada

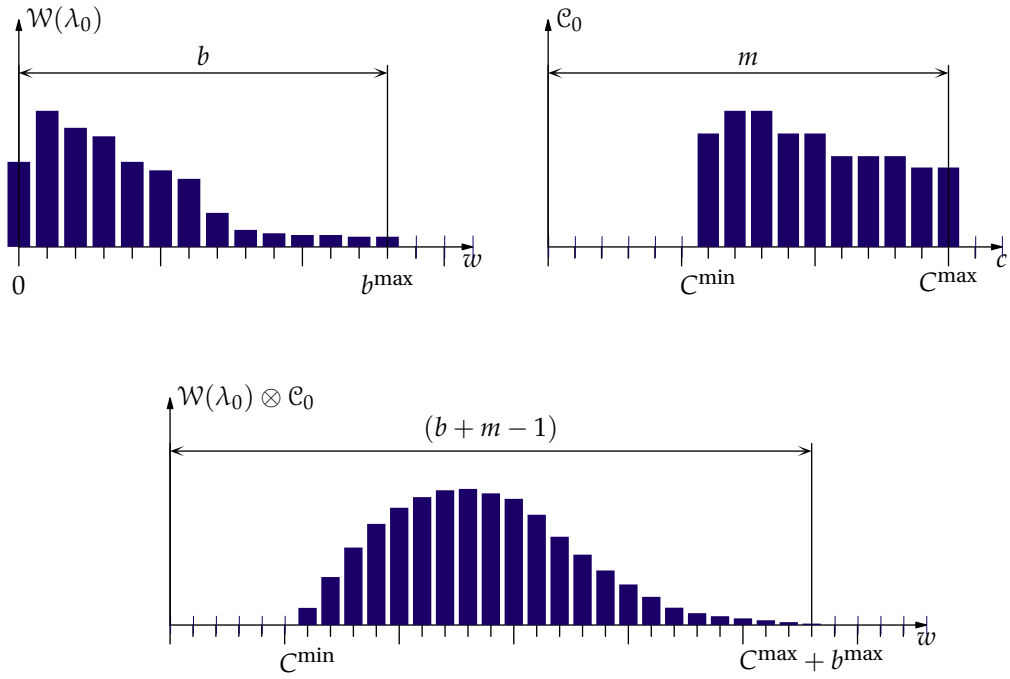


Figura 6.1.: Complejidad: paso "convolucionar"

una de las funciones que se convolucionan (ver anexo A). El resultado de esta convolución será una nueva PF definida por $(m_1 + m_2 - 1)$ puntos. Por tanto, en la primera convolución, entre la carga inicial y el primer trabajo, se realizarán $2bm$ operaciones, y el resultado tendrá $(b + m - 1)$ puntos. La figura 6.1 muestra gráficamente las dimensiones de la PF de la carga inicial, de un trabajo y de la convolución de ambas.

A continuación viene la operación "encoger". La cantidad a encoger es igual al intervalo temporal hasta la llegada del próximo trabajo. Esta distancia en promedio es $\bar{T}$, por lo que asumiremos que es necesario encoger $\bar{T}$ unidades. Como se puede observar en la figura 6.2, esta operación requerirá en el peor de los casos $\bar{T}$ operaciones (sumas), y un desplazamiento que asumiremos no tiene coste alguno[1]. El resultado será una nueva PF con $(b + m - 1 - \bar{T})$ puntos.

Observar que si $\bar{T} = m - 1$ el resultado tendría de nuevo b puntos, es decir no se aumentaría el tamaño del PF de la carga como resultado de la operación "convolucionar y encoger". Por otro lado, si $\bar{T} > m - 1$, el tamaño de la carga se iría reduciendo hasta llegar eventualmente a tener un solo punto (carga cero con probabilidad uno). En este sentido la fórmula no es correcta, puesto que puede dar un resultado negativo. La expresión correcta sería decir que la nueva longitud es $\max(1, (b + m - 1 - \bar{T}))$. Sin embargo, por no complicar la notación, asumiremos

[1] Puede implementarse como una simple reasignación de punteros

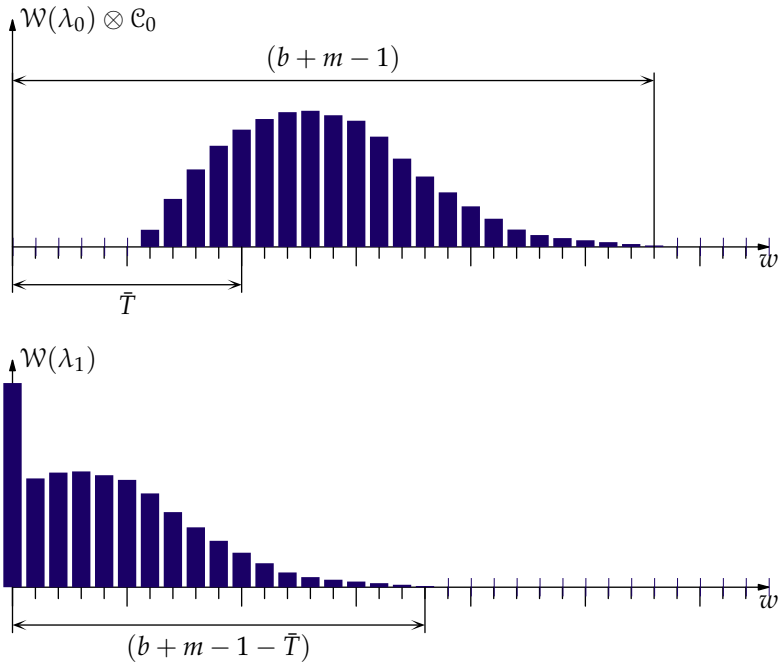


Figura 6.2.: Complejidad: paso “encoger”

que $\bar{T} < m-1$, y en este caso cada nueva iteración del método “convolucionar y encoger” produce un resultado con más puntos que la anterior, por lo que el coste computacional de cada nueva iteración irá creciendo.

En realidad, la condición $\bar{T} < m-1$ implica que $\bar{T} < C^{\max}$, y por tanto $U^{\max} = C^{\max}/\bar{T} > 1$. Por tanto, la hipótesis $\bar{T} < m-1$ se cumple al analizar sistemas con $U^{\max} > 1$, que son el tipo de sistemas en que estamos interesados. En este caso cada nueva iteración incrementa el número de puntos en $(m-1-\bar{T})$ puntos. En la iteración i-ésima tendremos que convolucionar una función definida por $(b+(i-1)(m-1-\bar{T}))$ puntos, que proviene de la iteración anterior, con otra de m puntos que es el trabajo i-ésimo. Seguidamente se realizará una operación “encoger” de $\bar{T}$ puntos.

Si denotamos por c_{W_j} al coste computacional (número de operaciones requeridas) de los j pasos necesarios para llegar al trabajo Γ_j y determinar así $\mathcal{W}(\lambda_j)$, este coste será por tanto:

$$c_{W_j} = \sum_{i=1}^{j} \left(2(b+(i-1)(m-1-\bar{T}))m + \bar{T} \right)$$

Resolviendo el sumatorio y operando, obtenemos:

$$\begin{aligned} c_{W_j} = j^2m^2 - jm^2 - j^2m(1+\bar{T}) \\ + jm(1+2b+\bar{T}) + j\bar{T} \end{aligned} \tag{6.1}$$

Si en esta ecuación hacemos $\bar{T} = m-1$, lo que significaría que $U^{\max} = 1$, se tiene el caso más favorable, en el que el tamaño de la carga se mantiene constante (por término medio) con cada nueva iteración. Para este caso c_{W_j} saldría igual a $j((2b+1)m-1)$, que como puede verse es del orden de $O(jm)$, para un tamaño de b fijo. El coste es proporcional al producto jm , es decir, del índice del trabajo que queremos calcular por el tamaño de las PF de los trabajos m.

El caso más desfavorable sería un caso límite en el cual los trabajos llegan tan juntos que la operación "encoger" nunca encoge realmente, y el tamaño de la carga siempre aumenta. Haciendo $\bar{T} \to 0$ obtenemos el coste para este caso que resulta ser $j^2m^2 - jm^2 - j^2m + jm(1+2b)$. Esta expresión es del orden de $O(j^2m^2)$, por tanto en este caso el coste es proporcional al *cuadrado* del producto jm. Por consiguiente, incluso para el peor caso, llegamos a la conclusión:

$$c_{W_j} = O(j^2m^2) \tag{6.2}$$

Coste computacional del método "partir, convolucionar y juntar"

La PF de la carga en el instante λ_j, obtenida por "convolucionar y encoger", constará de $(b + j(m-1-\bar{T}))$ puntos, de acuerdo con lo dicho anteriormente. A partir de ella, el primer paso es obtener $\mathcal{R}_j^{\langle 0\rangle}$, para lo cual hay que convolucionar dicha PF con la del tiempo de ejecución del trabajo, que se compone de m puntos. El coste de esta operación será $2(b + j(m-1-\bar{T}))m$ y el resultado constará de $(b + j(m-1-\bar{T}) + m - 1)$ puntos.

Analicemos el coste del método "partir, convolucionar y juntar" para la primera iteración:

- **Partir** implica reservar los primeros puntos de $\mathcal{R}_j^{\langle 0\rangle}$. Esta operación no tiene coste, pues no requiere operaciones de punto flotante.

  El número de puntos "reservados" será en promedio $\bar{T}+1$, para la primera iteración. La cola que queda, por tanto tendrá una longitud $(b + j(m-1-\bar{T}) + m - 1 - (\bar{T}+1))$ para la primera iteración.

- **Convolucionar**. La convolución se realiza entre la cola y el siguiente trabajo. El número de operaciones requeridas será

$$2(b + j(m-1-\bar{T}) + m - 1 - (\bar{T}+1))m$$

El resultado de esta convolución tendrá un número de puntos igual a la suma de las longitudes de las funciones convolucionadas, menos uno. Por tanto:

$$b + j(m - 1 - \bar{T}) + 2m - 1 - (\bar{T} + 1) - 1$$

- **Juntar**. Los $(\bar{T} + 1)$ puntos reservados al principio, se juntan con los obtenidos de la convolución. El número total de puntos del resultado será por tanto

$$b + j(m - 1 - \bar{T}) + 2m - 2$$

Y lo mismo se repite para la iteración siguiente. De lo anterior se deduce fácilmente que cada nueva iteración incrementa el número de puntos de $\mathcal{R}_j$ en $(m - 1)$ puntos. Así pues, para realizar el cálculo de $\mathcal{R}_j^{\langle i \rangle}$ es necesario:

- Tomar la PF de $\mathcal{R}_j^{\langle i-1 \rangle}$, que tiene $(b + j(m - 1 - \bar{T}) + i(m - 1))$ puntos, y reservar los primeros $i\bar{T} + 1$ puntos.

  La cola resultante tiene $(b + j(m - 1 - \bar{T}) + i(m - 1 - \bar{T}) - 1)$ puntos.

- Convolucionar dicha cola con el trabajo siguiente, definido por m puntos. El número de operaciones a realizar para esta convolución será:

$$2(b + (j + i)(m - 1 - \bar{T}) - 1)m$$

  Y el resultado constará de $(b + (j + i)(m - 1 - \bar{T}) - 1 + m - 1)$ puntos.

- Juntar a este resultado los $i\bar{T} + 1$ puntos reservados. El resultado final tendrá $(b + j(m - 1 - \bar{T}) + (i + 1)(m - 1))$ puntos.

El proceso anterior debe repetirse, en el peor de los casos, tantas veces como trabajos lleguen en el intervalo $[\lambda_j, \lambda_j + D_j]$. Llamemos k_j al número promedio de trabajos que llegan en dicho intervalo. Este valor puede calcularse como el cociente $D_j / \bar{T}$, y en general es diferente para cada trabajo, puesto que depende de su plazo D_j. En lugar de considerar un k_j para cada trabajo, simplificaremos el estudio de complejidad definiendo un k "global", que sea el valor promedio de todos los k_j. Es decir[2]:

$$k = \frac{1}{n} \sum_{j=1}^{n} k_j \quad \text{donde } k_j = D_j / \bar{T}$$

[2] Más adelante daremos una fórmula para calcular k en sistemas compuestos por tareas periódicas, y veremos que k depende de la relación entre plazos y periodos y también del número de tareas en el sistema.

Con esta aproximación podemos decir que, en promedio, un trabajo arbitrario del sistema será expulsado k veces, por lo que el método "partir, convolucionar y juntar" deberá realizar k iteraciones. Esto conlleva un número de operaciones igual a:

$$c_{R_j} = \sum_{i=0}^{k} 2(b + (j+i)(m-1-\bar{T}) - 1)m$$

Resolviendo el sumatorio y simplificando, obtenemos:

$$c_{R_j} = m(k+1)(2b - 2 + (2j+k)(m-1-\bar{T})) \tag{6.3}$$

Para el caso más favorable en el que $U^{\max} = 1$, y por tanto $\bar{T} = m-1$, la expresión anterior se reduce a $2m(b-1)(k+1)$, que como se observa tiene una complejidad $O(mk)$, para un b fijo. Es decir, el coste computacional es proporcional al producto del tamaño de las tareas por el número de interferencias. Para el caso más desfavorable en que las tareas llegan muy juntas, haciendo tender $\bar{T} \to 0$ en la ecuación (6.3) obtenemos un polinomio que es $O(m^2k^2)$, y por tanto en este caso el coste computacional es proporcional al *cuadrado* del producto mk.

Coste total de analizar los n trabajos del sistema

Suponiendo que la carga inicial ya ha sido obtenida por análisis Markoviano o por alguna aproximación (más adelante estudiaremos el coste computacional de esto), para resolver el tiempo de respuesta de un trabajo Γ_j basta realizar j iteraciones del método "convolucionar y encoger", seguido de k iteraciones del método "partir, convolucionar y juntar". El coste de lo primero es c_{W_j}, dado en la eq. (6.1), y el coste de lo segundo es c_{R_j}, dado en la eq. (6.3). Este análisis hay que repetirlo para cada uno de los n trabajos del hiperperiodo, por lo que parece obvio que el coste total sería:

$$c = \sum_{j=1}^{n} (c_{W_j} + c_{R_j})$$

Sin embargo hay un detalle sutil que nos permite afinar más la fórmula anterior. Y es que en esa fórmula se asume implícitamente que para calcular la carga existente a la llegada de cada trabajo, recomenzamos siempre el método "convolucionar y encoger" desde el inicio del hiperperiodo. Es decir, no reaprovechamos los resultados obtenidos para trabajos ya calculados. Esto será así sólo si el primer trabajo es basal y los restantes son no-basales. Esta situación tan pesimista nunca se dará en la práctica. De todas formas, si resolvemos la fórmula anterior, obtendremos el coste total para este caso pesimista. Como resultado se obtiene la

ecuación (6.4), que es demasiado compleja y trataremos de simplificarla a continuación.

$$c = n^3 \frac{m^2 - m(1+\bar{T})}{3} + n^2 \left(m^2(1+k) + m\big(b - 1 - k(\bar{T}+1) - \bar{T}\big) + \bar{T}/2 \right) + n \Bigg(\bar{T}/2 + m\big(k^2(-1-\bar{T}) + 2k(b-2-\bar{T}) + 3b - 8/3 - 2\bar{T}/3\big) + m^2(k^2 + 2k + 2/3) \Bigg) \tag{6.4}$$

Hipótesis simplificadoras La fórmula anterior, además de ser un tanto farragosa, depende de demasiados parámetros y es difícil extraer conclusiones útiles de ella. Sería interesante poder dejar la expresión del coste únicamente en función de n y m. Esto puede lograrse planteando dos casos extremos, uno aplicable a sistemas con baja utilización y otro aplicable a sistemas con muy alta utilización. El primer caso será el de menor coste computacional, y nos proporciona un límite inferior a la complejidad del algoritmo. El segundo caso, que es el más costoso computacionalmente, nos proporciona un límite superior a dicha complejidad.

- **Sistemas con baja utilización**. Si el sistema a estudiar tiene $U^{\max} < 1$, se reduce la complejidad del análisis debido a varios factores:

  - No es necesario análisis Markoviano, y la "carga inicial" en el hiperperiodo estacionario es cero.
  - La distancia promedio entre activaciones es mayor que la carga promedio generada en cada activación, esto es, $\bar{T} \geq m - 1$. Esto, como ya se ha visto, tiene como consecuencia que la PF de la carga inicial no crece con cada iteración del algoritmo "convolucionar y encoger".
  - Hay bajas probabilidades de que un trabajo sea expulsado por otro, esto es, k tomará valores bajos.

  Estas condiciones nos permiten eliminar algunas variables de la fórmula anterior, gracias a las siguientes ecuaciones:

  - $b = 1$ (debido a que la carga inicial estacionaria es cero, y por tanto representada por una PF de un solo punto).
  - $\bar{T} = m - 1$ en el peor caso.

- Si hacemos la suposición de que las tareas tienen plazos iguales a sus periodos, se puede demostrar (ver página 207) que k es igual al número de tareas (no de trabajos). Es decir $k = N$.

Llevando estas sustituciones de variables a la ecuación (6.4), obtenemos un límite inferior al coste, que ya depende sólo de n y m:

$$c \geq (n^2 + n)\frac{3m-1}{2} = O(n^2 m) \tag{6.5}$$

- **Sistemas con muy alta utilización**. En estos sistemas $U^{\max} > 1$, y por tanto requieren un análisis Markoviano. No obstante, en este punto supondremos simplemente que la distribución de la carga inicial en el hiperperiodo estacionario ya ha sido obtenida de algún modo, y nos centraremos únicamente en el coste de obtener los tiempos de respuesta a partir de ella.

  Haremos las siguientes hipótesis (pesimistas) para tratar de dejar la ecuación (6.4) tan sólo en función de n y m:

  - La PF de la carga inicial en el hiperperiodo estacionario se compone de un máximo de nm puntos. Esto es, $b = mn$. Téngase en cuenta que en realidad esta función consta de un número infinito de valores, pero para poder introducirla en la herramienta de análisis debe ser truncada en algún punto. Estamos suponiendo simplemente que el punto de truncación es a lo sumo nm.

    Esta hipótesis se puede justificar en la forma siguiente. Si cada trabajo genera como máximo $m-1$ unidades de tiempo de ejecución, y hay n trabajos activados en un hiperperiodo, el producto mn es mayor que la máxima carga que puede generarse en un hiperperiodo. La probabilidad de que la carga inicial sea superior a mn por tanto ha de ser muy baja, de modo que los puntos que estamos perdiendo al truncar en mn son muy próximos a cero.
  - Los trabajos llegan muy juntos, de modo que hacemos $\bar{T} \to 0$
  - Los plazos de los trabajos son muy largos, del orden de la longitud del hiperperiodo. Por tanto cada trabajo puede ser expulsado a lo sumo por otros n trabajos, es decir $k = n$. Obsérvese que esta hipótesis deja de ser cierta si los plazos *relativos* pudieran tener una duración mayor que el hiperperiodo, pero considero que un sistema así es poco realista.

  Esto nos permite sustituir b por mn, $\bar{T}$ por cero, y k por n en la ecuación (6.4), lo que nos genera el siguiente límite superior al coste computacional:

$$c \leq n^3(\frac{16m^2-7m}{3}) + n^2(6m^2-5m) + n(\frac{2m^2-8m}{3}) = O(n^3 m^2) \tag{6.6}$$

Las figuras 6.3 y 6.4 muestran una representación gráfica de las ecuaciones (6.5) y (6.6), respectivamente, en las que en el eje de abscisas se hace variar n y se representan curvas para diferentes valores de m. Observar que la escala es logarítmica.

En la figura 6.5 se representan juntas las ecuaciones (6.5) y (6.6), para un caso en el que $m = 20$, haciendo variar únicamente n. Puesto que estas ecuaciones representan los límites inferior y superior respectivamente del coste computacional, cualquier otro sistema con $m = 20$ requerirá un número de operaciones comprendido entre ambas curvas.

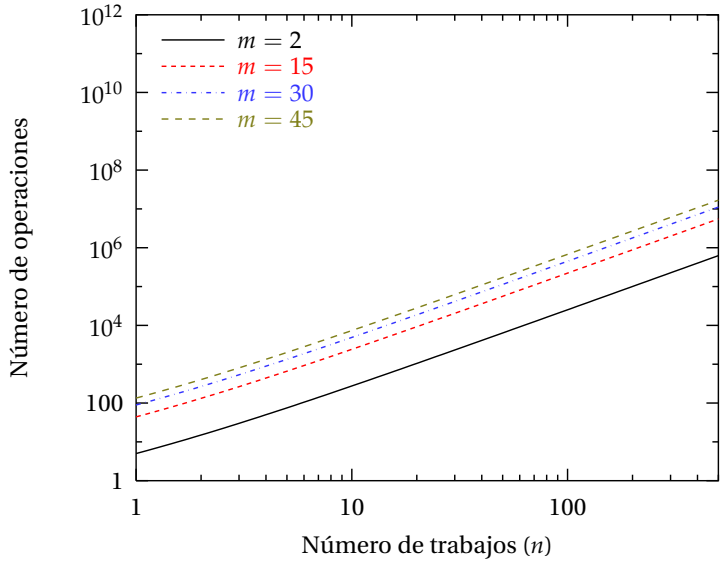


Figura 6.3.: Variación del coste en función de los parámetros m y n, para sistemas con baja utilización

Observar que en el caso $\bar{T} = m - 1$ y $\bar{T} \to 0$ son los extremos de un espectro. En el primer caso el sistema tiene $U^{\max} = 1$, y en el segundo en cambio $U^{\max} \to \infty$. Puede resultar de interés saber cómo influyen otros valores intermedios de $U^{\max}$ en el coste computacional. Para ello basta sustituir $\bar{T}$ por $(m-1)/U^{\max}$ en la fórmula (6.4) para tener una expresión que relaciona el coste computacional con la carga máxima del sistema[3].

En la figura 6.6 se representa de forma gráfica la variación del coste en función de $U^{\max}$, dejando fijos los restantes parámetros del sistema. Vemos que aunque $U^{\max}$ influye, su influencia es cada vez menor a medida que crece, pues está limitada por una asíntota horizontal, que viene dada por el valor de la ecuación (6.4) para $\bar{T} \to 0$.

[3] En realidad la expresión sólo sería válida para $U^{\max} > 1$, puesto que para valores menores el método "convolucionar y encoger" iría reduciendo el número de puntos en la PF de la carga, hasta dejarlo eventualmente en 1.

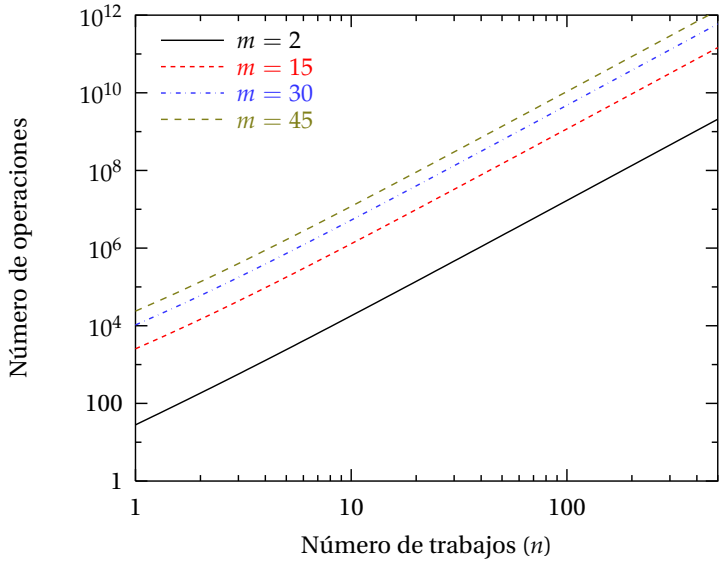


Figura 6.4.: Variación del coste en función de los parámetros m y n, para sistemas con alta utilización

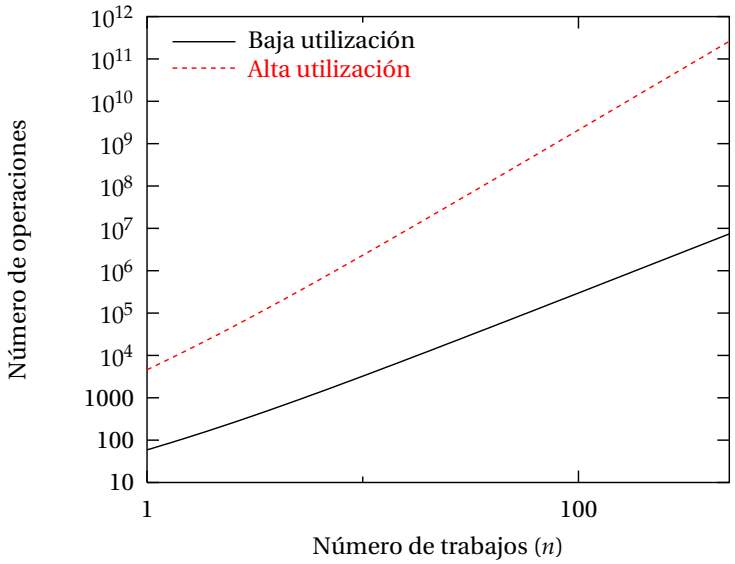


Figura 6.5.: Comparación entre el coste de resolver un sistema con baja utilización y otro con alta. En ambos casos $m = 20$

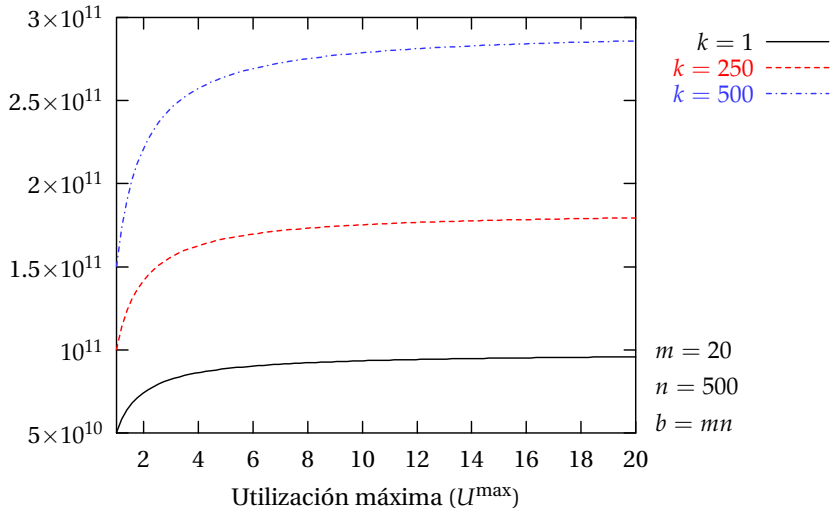


Figura 6.6.: Coste computacional en función de $U^{\max}$, para un sistema con 500 trabajos de tamaño 20, y diferentes valores de k

A continuación nos centraremos en estimar el coste computacional del cálculo de la distribución de la carga al principio del "hiperperiodo estacionario". Para ello disponemos de diferentes métodos, a saber: el análisis Markoviano exacto, basado en plantear un sistema de ecuaciones recurrentes del que se obtienen los infinitos puntos de la solución, el análisis Markoviano por truncación, y el método iterativo que se basa en repetir el método "convolucionar y encoger" a lo largo de varios hiperperiodos. Cada método tendrá un coste diferente.

Coste de encontrar la carga estacionaria mediante análisis Markoviano exacto

Este método requiere varios pasos, todos ellos muy costosos desde el punto de vista computacional:

1. Obtener la matriz de Markov $\mathbf{P}$ que define el proceso de la carga. Como se vio, solo es necesario obtener sus r primeras columnas, siendo r la máxima cantidad de tiempo libre que puede aparecer en un hiperperiodo. Llamaremos c_P al coste de obtener esta matriz $\mathbf{P}$.

2. Plantear una matriz $\mathbf{A}$ de recurrencia y diagonalizarla. El coste de diagonalizar una matriz cuadrada es del orden del cubo de su número de filas. Vimos que la matriz $\mathbf{A}$ tenía m_r filas, siendo m_r el número de elementos de la columna r-ésima de $\mathbf{P}$. Llamaremos c_A al coste de diagonalizar $\mathbf{A}$.

3. Plantear y resolver un sistema con $r + m_r$ ecuaciones y otras tantas incógnitas. En general el coste de resolver un sistema es del orden del cubo del número de ecuaciones. Llamaremos c_S al coste de resolver este sistema.

Calculemos pues cuál es el coste de cada una de estas fases, intentando poner todas ellas en función de los parámetros m, n y $\bar{T}$ (en este caso el factor k no juega ningún papel).

Obtención de la matriz de Markov El cálculo de una columna i de esta matriz se obtiene aplicando el método "convolucionar y encoger", partiendo de una carga inicial i, a lo largo de un hiperperiodo, es decir, de n trabajos. La carga inicial viene dada por un solo punto de probabilidad 1 para la abscisa i. Por tanto el tiempo necesario para obtener una columna de $\mathbf{P}$ viene dado por la ecuación 6.1, haciendo $b = 1$ y $j = n$. Puesto que hay que calcular r columnas, el coste total será r veces lo anterior, es decir:

$$c_P = r\left(n^2(m^2 - m(\bar{T}+1)) - n(m^2 - m(\bar{T}+3) + \bar{T})\right) \tag{6.7}$$

Resulta más significativo poner la ecuación en función de $U^{\max}$, en lugar de $\bar{T}$. Es posible hacerlo, ya que $U^{\max} = C^{\max}/\bar{T}$, y ya que $C^{\max} = m - 1$, resulta que $U^{\max} < m/\bar{T}$ y por tanto $\bar{T} < m/U^{\max}$. Llevando esto a la ecuación anterior y operando, queda:

$$c_P < rn^2\left(m^2 - m - \frac{m^2}{U^{\max}}\right) - rn\left(m^2 - 3m - \frac{m^2+m}{U^{\max}}\right) \tag{6.8}$$

Podemos hacer que r desaparezca de esta expresión, si tenemos en cuenta que representa máxima cantidad de tiempo libre que puede aparecer en un hiperperiodo. Esta cantidad nunca podrá ser mayor que la longitud del hiperperiodo, por lo que $r < n\bar{T}$. Y ya que por otro lado $\bar{T} = m/U^{\max}$, y estamos en el caso $U^{\max} > 1$, concluimos finalmente que $r < nm$. Por tanto:

$$c_P < n^3\left(\frac{m^3 - m^2}{U^{\max}} - \frac{m^3}{U^{\max 2}}\right) + n^2\left(\frac{m^3 + m^2}{U^{\max 2}} - \frac{m^3 - 3m^2}{U^{\max}}\right) \tag{6.9}$$

En esta expresión puede verse que, para un $U^{\max}$ dado y mayor de uno, el coste es del orden $O(m^3n^3)$. Recordemos que si $U^{\max} < 1$ no es necesario plantear el análisis Markoviano, pues en ese caso basta analizar el primer hiperperiodo completo. Así pues, el coste de obtener la matriz $\mathbf{P}$ es proporcional al cubo del producto del número de trabajos en el hiperperiodo por el tamaño de estos trabajos (en el peor de los casos).

Obtención y diagonalización de $\mathbf{A}$ El planteamiento de la matriz $\mathbf{A}$ apenas tiene coste computacional, pues su estructura consta exclusivamente de unos y ceros salvo en su última fila, y para ésta los coeficientes se obtienen directamente de la columna r de $\mathbf{P}$. La única computación que es necesario realizar es dividir todos

los coeficientes de la columna r de $\mathbf{P}$ por uno de sus coeficientes. Esto requiere m_r divisiones.

Por otro lado m_r es el número de elementos de la columna r-ésima de $\mathbf{P}$. Si recordamos, esta columna se había obtenido convolucionando los n trabajos que llegan en el hiperperiodo. Ya que cada convolución añade $m-1$ puntos, el resultado final constará de $m+(n-1)(m-1) = mn-n+1$ puntos. Por tanto $m_r < mn$, de modo que el número de operaciones necesario para construir $\mathbf{A}$ es menor que el producto mn.

Para su diagonalización se requiere un número de operaciones del orden de su dimensión elevada al cubo. Ya que su dimensión es también m_r, el coste total de obtener y diagonalizar $\mathbf{A}$ será:

$$c_A < mn + (mn)^3 \tag{6.10}$$

Se comprueba que el coste de este paso es también del orden de $O(n^3m^3)$.

Planteamiento y solución del sistema de ecuaciones Una vez diagonalizada $\mathbf{A}$, se plantea un sistema con m_r ecuaciones obtenidas de las primeras filas de $\mathbf{P}$, más r ecuaciones adicionales obtenidas de hacer cero ciertos coeficientes para que la suma de la solución no sea infinita. El sistema constará por tanto de $m_r + r$ ecuaciones, por lo que su resolución tiene un coste del orden de $O((m_r+r)^3)$. Como ya vimos, $r < mn$ y también $m_r < mn$, por lo que el número total de ecuaciones es menor de $2mn$. Se deduce que la complejidad de resolver este sistema de ecuaciones es del orden de

$$c_S = O\big((2mn)^3\big) = O(m^3n^3) \tag{6.11}$$

Coste total del análisis Markoviano exacto El coste total del análisis Markoviano, por tanto, será la suma de los tres costes anteriores, es decir:

$$c_M = c_P + c_A + c_S \tag{6.12}$$

Puesto que, como hemos visto, cada uno de ellos es del orden de $O(m^3n^3)$, su suma también lo será.

Coste de encontrar la carga estacionaria mediante truncación

Obviamente, la complejidad de esta aproximación depende del punto de truncación. Supongamos que truncamos $\mathbf{P}$ a la dimensión $mn \times mn$, con vistas a obtener una solución de longitud mn, que como ya dijimos es una longitud suficientemente precisa en la práctica.

Ya que, como vimos, $r < mn$, para truncar la matriz a la dimensión mn, será necesario de todas formas calcular las r primeras columnas, lo mismo que para el análisis exacto. El coste de esto ya ha sido calculado, en la eq. (6.9) y vimos que era del orden de $O(m^3n^3)$

Una vez truncada la matriz a la dimensión mn, basta con obtener su vector propio. Esto requiere $O((mn)^3)$ operaciones. El coste total del método de truncación será por tanto:

$$c_T = O(m^3n^3) + O(m^3n^3) = O(m^3n^3) \tag{6.13}$$

Por tanto, aunque el método de truncación requiere menos tiempo en general que el análisis exacto, porque evita un paso de complejidad $O(m^3n^3)$, desde el punto de vista del análisis asintótico de la complejidad es equivalente. En realidad, el método de truncación tarda una fracción de lo que tardaría el método exacto, pero esta fracción es fija, no depende de m ni n y por eso no aparece diferencia en la notación O-grande.

El método de truncación, por otra parte, tiene una ventaja sobre el exacto, y es que es menos sensible a los errores numéricos introducidos por la representación de coma flotante en un computador, como ya se comentó.

Coste de encontrar la carga estacionaria por aproximación iterativa

El método iterativo consiste simplemente en aplicar el método "convolucionar y encoger" a lo largo de varios hiperperiodos, hasta que el resultado en un hiperperiodo sea muy similar al del hiperperiodo anterior. No requiere por tanto la construcción de ninguna matriz de Markov ni la resolución de ningún sistema de ecuaciones.

Si el método converge tras I iteraciones, cuando lleguemos al punto de la convergencia habremos aplicado el método "convolucionar y encoger" nI veces. Ya que la carga inicial de la que partimos es cero, viene representada por un solo punto. De modo que el coste computacional de todas estas iteraciones viene dado por la fórmula (6.1), haciendo $j = nI$ y $b = 1$, de lo que resulta:

$$c_I = I^2n^2(m^2 - m(\bar{T} + 1)) - In(m^2 - m(3 + \bar{T}) - \bar{T}) \tag{6.14}$$

Esta cantidad, para un $\bar{T}$ fijo, es del orden de $O(I^2n^2m^2)$, de modo que si el número de iteraciones no es muy elevado, este método es muy ventajoso con respecto al cálculo exacto e incluso con respecto a la truncación. Sólo cuando el número de iteraciones necesarias sea del orden de $\sqrt{mn}$ o más, el método iterativo tiene mayor coste que los otros métodos. El inconveniente es que no sabemos de antemano cuántas iteraciones serán necesarias para que el método converja, es decir, el valor de I. En general, el método iterativo converge rápido y es mucho

más eficiente computacionalmente que la truncación. Además, no presenta los problemas numéricos del método exacto.

6.1.3. Conclusiones del análisis de complejidad

A lo largo de las fórmulas anteriores hemos visto que el tiempo del cómputo parece fuertemente influenciado por los siguientes parámetros del sistema:

- El número total de **trabajos en un hiperperiodo** (n). La dependencia de este parámetro es $O(n^3)$, y por tanto es la principal causa de incremento del coste. De aquí se deduce que si el sistema tiene muchas tareas con periodos co-primos, n será muy grande y por tanto el análisis tendrá un gran coste. Si por el contrario los periodos de las tareas se eligen de forma que sean armónicos, el número de trabajos por hiperperiodo será menor, lo que ayudará a reducir el coste del análisis.

- El **tamaño de la función de probabilidad** de cada trabajo (m). La dependencia de este parámetro es $O(m^2)$ si se usa la aproximación iterativa para el cálculo de la carga estacionaria, y $O(m^3)$ si se usa el análisis Markoviano exacto o truncado. Por tanto es otro factor crucial. Si el coste del análisis resultara prohibitivo para un sistema dado, habría que intentar replantear el problema reduciendo el número de puntos en las PF de los tiempos de ejecución. Para ello basta remuestrear el perfil de ejecución en un número menor de puntos. Obviamente esto disminuirá la precisión del análisis, pero siempre se puede hacer de modo que el nuevo análisis sea más pesimista y por tanto se esté del lado de la seguridad (ver anexo C.4). En cualquier caso, incluso un valor tan pequeño como $m = 2$ representa una disminución en el pesimismo con respecto al análisis clásico, en el que $m = 1$ puesto que sólo se toma en cuenta un posible tiempo de ejecución: el peor.

- El **tiempo medio entre activaciones** ($\bar{T}$) juega un papel importante en el coste del análisis. Este parámetro representa si los trabajos llegan muy juntos o muy separados. Como vimos $\bar{T} < m/U^{\max}$, por lo que está directamente relacionado con la **utilización máxima del sistema**. A medida que aumenta $U^{\max}$, para una m dada, $\bar{T}$ disminuye y esto causa que el análisis tenga mayor coste.

  El caso con $U^{\max} \leq 1$ es el de menor coste computacional, puesto que, de un lado, no requiere análisis Markoviano (la carga de régimen estacionario se obtiene ya al final del primer hiperperiodo), y de otro lado, el método "convolucionar y encoger" tiene una complejidad constante, independiente del trabajo en consideración, ya que el tamaño de la PF de la carga no crece en cada nueva iteración.

A medida que $\bar{T} \to 0$, $U^{\max}$ va creciendo y el coste del algoritmo se incrementa. Pero incluso en el caso límite en que $\bar{T} = 0$, el coste está acotado por $O(m^3 n^3)$, o por $O(m^3 n^2)$ si usamos la aproximación iterativa.

- Los **plazos de las tareas** influyen en el número de interferencias (k) que deben tenerse en cuenta para computar la probabilidad de que un trabajo pierda su plazo. Cuanto menores sean los plazos, menos trabajos hay que considerar, y más rápido será el algoritmo "partir, convolucionar y juntar".

El valor de la utilización promedio, $\bar{U}$, no parece jugar ningún papel en el coste computacional puesto que no aparece en las fórmulas.

Estrictamente esto es así, ya que desde el punto de vista teórico, es posible diseñar dos sistemas que tengan el mismo valor de $\bar{T}$ (las tareas tienen los mismos periodos) y de m (y el mismo tiempo máximo de ejecución) y que sin embargo tengan diferentes $\bar{U}$. Basta hacer que los tiempos medios de ejecución de cada tarea sean diferentes, cambiando la "forma" de su distribución, aunque mantengamos el número de puntos m. Digamos por ejemplo que el tiempo promedio de ejecución es una fracción f del tiempo máximo, de modo que $\bar{C}_j = f C_j^{\max}$. Cambiando el valor de f podemos obtener un sistema nuevo con diferente $\bar{U}$ pero idénticos valores en los restantes parámetros. El análisis de estos dos sistemas tendría el mismo coste computacional, a pesar de que tienen diferente $\bar{U}$, debido a que tienen los mismos valores de $\bar{T}$ y m.

En la práctica, en cambio, la situación suele ser la inversa, es decir, la mayoría de las tareas tienen una forma similar en sus funciones de distribución del tiempo de ejecución, es decir, un valor de f parecido. Por esto, un sistema que tenga un mayor valor de $\bar{U}$ suele tener también un mayor valor de $U^{\max}$. En este sentido la utilización media del sistema sí influye en el coste computacional de su análisis.

Por otro lado, hemos visto que $\bar{U}$ es un parámetro clave que determina la existencia o inexistencia de una solución estacionaria. Si $\bar{U} \geq 1$ esta solución no existe, mientras que si $\bar{U} < 1$ sí. Cabe esperar por tanto que a medida que $\bar{U}$ se acerca a 1, la solución estacionaria tarde más tiempo en alcanzarse. En este sentido $\bar{U}$ influirá sobre el número de iteraciones que debe realizarse en la aproximación iterativa. También influirá en la elección del punto de truncación de la aproximación por truncación.

6.2. Medidas experimentales

Como comprobación empírica de que el coste del análisis crece polinómicamente con el tamaño del sistema a analizar, se han generado un gran número de sistemas sintéticos y se han analizado con una herramienta que implementa el análisis estocástico propuesto, para medir el tiempo invertido por la herramienta en función de los diferentes parámetros del sistema.

6.2.1. Modelo para los experimentos

Puesto que la motivación de los experimentos es únicamente medir el tiempo que se requiere para resolver sistemas de diferentes tamaños, los parámetros que interesa controlar son los que afectan al coste computacional y que ya han sido identificados en la sección 6.1.3, página 195. En este sentido, la forma de la PF de los tiempos de ejecución de cada trabajo no es relevante, pero sí lo es el número de puntos m que la define. Tampoco son relevantes los periodos de cada tarea, sino el número total de trabajos que habrá en un hiperperiodo y la separación promedio entre los instantes de llegada, etc. Así pues definiremos un modelo diferente de sistema, con el objetivo de que sea sencillo variar sus parámetros y medir el tiempo requerido para su análisis, en una forma consistente con el análisis de complejidad previo.

El sistema se compondrá de una secuencia de n trabajos que llegan en instantes generados aleatoriamente siguiendo una cierta distribución parametrizada por su valor medio, $\bar{T}$. Cada uno de los trabajos trae asociada una PF definida por m puntos que representa la PF de su tiempo de ejecución. La PF se almacena en un vector de m elementos, siendo el elemento de índice 0 la probabilidad de que el tiempo de ejecución tome su valor mínimo $C^{\min}$ y el elemento de índice $(m-1)$ la probabilidad de que tome su valor máximo $C^{\max}$. Se asume que todos los trabajos tienen la misma PF de tiempo de ejecución. Se asume también que todos los trabajos tienen el mismo plazo relativo D.

La **especificación de un experimento** consistirá de las siguientes métricas, a partir de las cuales será posible generar automáticamente un sistema sintético que los cumpla:

- Utilización máxima del sistema $U^{\max}$.
- Utilización promedio del sistema $\bar{U}$.
- Número de trabajos a generar n.
- Tamaño m de la PF del tiempo de ejecución de cada trabajo y valor de $C^{\min}$. El valor de $C^{\max}$ se deduce de los anteriores, puesto que $C^{\max} = C^{\min} + m - 1$. Todos los trabajos tienen la misma PF, ya que su forma no es relevante.
- Número esperado de expulsiones por trabajo k.

6.2.2. Generación del sistema sintético

Partiendo de los parámetros anteriores, es posible generar un sistema sintético que se ajusta al modelo de "secuencia de trabajos" y que tiene los valores especificados para las métricas anteriores. Para ello basta seguir los pasos siguientes:

- Dado que $U^{\max} = C^{\max}/\bar{T}$ y que $C^{\max} = C^{\min} + m - 1$, es posible deducir cuál ha de ser el valor de $\bar{T}$ para que el sistema tenga la utilización máxima deseada:

$$\bar{T} = \frac{C^{\min} + m - 1}{U^{\max}}$$

  Una vez obtenido $\bar{T}$, podemos calcular cuál es la utilización mínima de este sistema $U^{\min} = C^{\min}/\bar{T}$.

- Se asume una distribución estadística conocida para los intervalos entre llegadas, por ejemplo la distribución beta (ver anexo C). Los parámetros de la distribución se eligen de modo que su media coincida con el $\bar{T}$ recién calculado.

  Se genera aleatoriamente una secuencia de n instantes de llegada, de modo que los intervalos entre llegadas sigan esta distribución. Estos n trabajos representan un hiperperiodo. Los hiperperiodos siguientes repetirán la misma secuencia de activaciones.

- A cada trabajo se le asigna un plazo relativo de valor $D = k\bar{T}$.

- Se asume una distribución estadística conocida para los tiempos de ejecución de las tareas, por ejemplo, la beta de nuevo. Aunque esta distribución no tiene influencia en el coste computacional, sí influye sobre la utilización media.

  El valor promedio del tiempo de ejecución $\bar{C}$, evidentemente estará a medio camino entre $C^{\min}$ y $C^{\max}$, por lo que podemos decir que existe un valor $h \in [0,1]$ tal que:

$$\bar{C} = C^{\min} + h(C^{\max} - C^{\min})$$

  De la expresión anterior, dividiendo por $\bar{T}$, se deduce que:

$$\bar{U} = U^{\min} + h(U^{\max} - U^{\min})$$

  lo que nos permite elegir h de tal modo que $\bar{U}$ tome cualquier valor deseado entre $U^{\min}$ y $U^{\max}$. Una vez obtenido este h, la PF de los tiempos de ejecución ya no es completamente libre, pues debe cumplir la relación $\bar{C} = C^{\min} + h(C^{\max} - C^{\min})$.

- Es necesario ahora crear una PF sintética de los tiempos de ejecución de los trabajos, que cumpla una serie de requisitos: debe estar formada por m puntos, variar sólo entre los valores $C^{\min}$ y $C^{\max}$, y su valor medio debe ser tal que $\bar{C} = C^{\min} + h(C^{\max} - C^{\min})$.

Para crear una PF así, resulta idóneo partir de la distribución beta (véase anexo C). Originalmente esta distribución es continua, y varía sólo entre 0 y 1 (por tanto tiene acotados sus valores mínimo y máximo). Su forma depende de dos parámetros habitualmente llamados α y β. Según los valores de estos parámetros, la distribución puede adoptar la forma de una distribución uniforme, una rampa, una especie de exponencial acotada, una campana simétrica, o una campana asimétrica. La media de la distribución es el valor $\alpha/(\alpha+\beta)$, por lo que puede tomar cualquier valor entre 0 y 1. En particular podemos hacer que su media tome el valor h antes calculado.

Basta ahora escalar horizontalmente esta función multiplicando por $(C^{\max} - C^{\min})$ y desplazarla sumando $C^{\min}$, para obtener una función de densidad, que después se puede muestrear en m puntos. De este modo se obtiene una PF sintética para los tiempos de ejecución de los trabajos del sistema (ver en el anexo C una discusión de cómo realizar dicho muestreo).

El sistema así construido estará formado de n trabajos, cada uno de ellos definido por m puntos como se deseaba. Además, los valores de $U^{\max}$ y $\bar{U}$ son los deseados. El sistema sintético así construido se suministra como entrada a la herramienta de análisis y se mide cuánto tarda ésta en hallar la PF de la carga en cada uno de los instantes de llegada, y la PF de los tiempos de respuesta para cada uno de los trabajos del sistema, asumiendo un caso pesimista (e imposible) en el que, para cada trabajo, todos los anteriores y todos los posteriores tienen mayor prioridad que él. De este modo hay que realizar el máximo de convoluciones.

Variando m, n, k y $U^{\max}$ podemos comprobar que el coste computacional sigue la evolución predicha en la sección anterior. $\bar{U}$ es un parámetro que en teoría no debería tener influencia si los restantes se dejan fijos.

En la sección siguiente se muestran los resultados de estos experimentos. Todos los experimentos fueron realizados con una herramienta programada en C por el autor, y ejecutados en un computador personal de gama media. Las características del *hardware* y del *software* utilizados fueron las siguientes:

- CPU: Pentium III, 933MHz, 256Kb cache
- Memoria principal: 512Mb
- Sistema operativo: linux, versión del *kernel* 2.4.8-26mdk
- Compilador: gcc 2.96, glibc 2.2.4-6mdk
- Generador de números aleatorios: "*Mersene Twister*" [Matsumoto y Nishimura, 1998], implementado en la *GNU scientific library*. Semilla del generador tomada del reloj del sistema (resolución de centésimas de segundo).

6.2.3. Resultados

En cada experimento se ha generado un sistema sintético que tenga los parámetros especificados, y se ha introducido este sistema en la herramienta, a la que se le ha pedido la PF de la carga total en el instante de activación de un trabajo Γ_j, y la PF del tiempo de respuesta de dicho trabajo. Es decir, la herramienta calcula las PFs de $\mathcal{W}(\lambda_j)$ y $\mathcal{R}_j$, siendo j un parámetro del experimento.

Se ha medido el tiempo que ha tardado la herramienta en obtener cada uno de estos PFs, y a continuación se muestra en forma de gráficos cómo han influido los diferentes parámetros (j, m, $U^{\max}$, $\bar{U}$, k) en estos tiempos.

Influencia de la utilización promedio $\bar{U}$

Manteniendo fijos todos los parámetros del experimento j, m, k y $U^{\max}$, pero variando la forma de las distribuciones de los tiempos de ejecución, podemos hacer que cada trabajo tenga el tiempo de ejecución promedio que deseemos. Por ejemplo, si damos una forma de campana fuertemente asimétrica y desplazada muy a la izquierda a los tiempos de ejecución de los trabajos, sus tiempos de ejecución promedio serán muy bajos en comparación a sus tiempos máximos (como suele ocurrir en la práctica). Si en cambio la hacemos asimétrica pero hacia la derecha, los tiempos medios serán cercanos al tiempo máximo.

Eligiendo la distribución beta para representar los tiempos de ejecución de los trabajos, tenemos libertad para elegir la forma de la misma (ver anexo C). Al alterar esta forma, por tanto, modificaremos la utilización promedio del sistema, ya que $\bar{U} = \bar{C}/\bar{T}$.

De acuerdo con el estudio de complejidad, esta variación de $\bar{U}$ no debería afectar al coste computacional, ya que lo que realmente afecta es la $U^{\max}$ que viene dada por m y $\bar{T}$. Verificamos experimentalmente que esto es así y los resultados se muestran en la gráfica 6.7, en la que hemos fijado los parámetros $n = 500$, $m = 35$, $k = 10$, $U^{\max} = 20$ y hemos hecho variar $\bar{U}$ entre 0,40 y 0,95. Se ha medido el tiempo de análisis necesario para determinar la PF del tiempo de respuesta del último trabajo (lo que requiere tomar en consideración los 500 previos). Como se esperaba, no se observa dependencia de $\bar{U}$, por lo que este parámetro ya no se tiene en cuenta en los restantes experimentos.

Influencia de la utilización máxima $U^{\max}$

Dejando fijos los valores de m y j, y variando $U^{\max}$, logramos que los trabajos lleguen cada vez más apretados. De acuerdo con el análisis de complejidad previo esto debería causar un mayor coste computacional a medida que aumenta $U^{\max}$. Sin embargo, también deberíamos observar que el incremento en el coste está

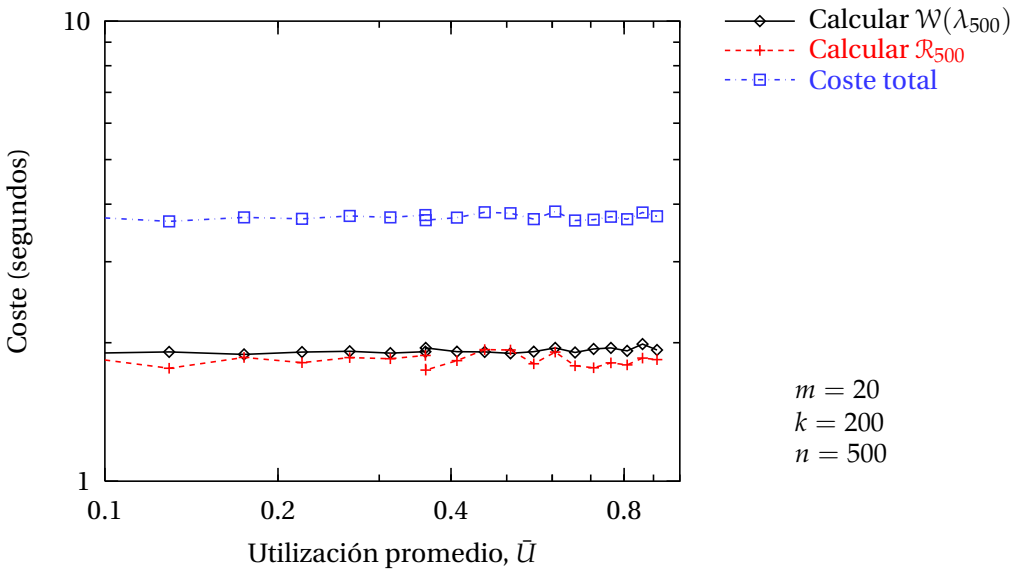


Figura 6.7.: Coste computacional en función de $\bar{U}$, dejando fijos los restantes parámetros.

acotado, puesto que incluso cuando $U^{\max} \to \infty$, esto es $\bar{T} \to 0$, obteníamos una expresión finita para el coste en función de m y n.

Efectivamente, la gráfica 6.8 muestra esta tendencia. La gráfica hace patente que por muy grande que sea la $U^{\max}$ para un sistema con 500 trabajos, de 20 puntos cada uno, el tiempo requerido para hallar el tiempo de respuesta del último trabajo (que es el que tiene mayor coste computacional) no excede los 4 segundos. Este tiempo incluye el coste de obtener la carga en el instante de su activación, pero partiendo de una carga inicial cero. Es decir, no incluye el coste del análisis Markoviano.

Coste del cálculo de la carga

Como ya se ha explicado en su momento, el método exacto para el análisis Markoviano no tiene apenas aplicación práctica, debido de un lado a su enorme complejidad computacional y de otro a la gran inestabilidad que presenta ante los errores numéricos inherentes al cálculo con computador. La solución exacta es interesante tan sólo desde el punto de vista teórico, ya que se muestra que la solución siempre tiene una forma exponencial. En esta sección experimental no la tendremos en cuenta.

Entre los dos métodos de aproximación presentados (truncación e iteración), el más eficiente y sencillo de implementar es el de la iteración, y en la práctica será el preferido. Por tanto tampoco analizaremos experimentalmente el coste computacional del método de truncación.

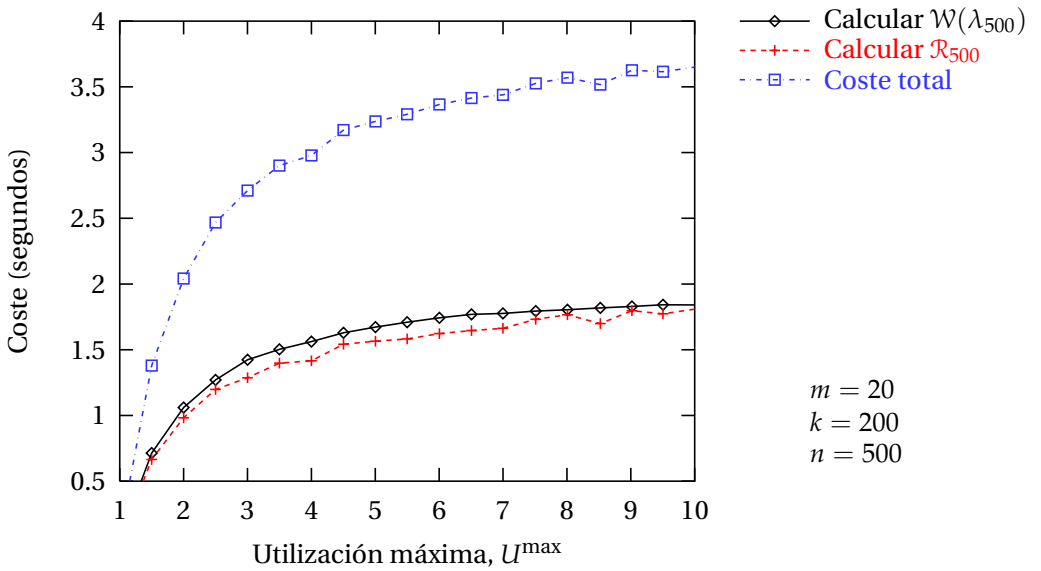


Figura 6.8.: Coste computacional de hallar la carga y el tiempo de respuesta del trabajo de índice $j = 500$, en función de $U^{\max}$, para sistemas con $m = 20$

Nos centraremos por tanto en el método iterativo para el cálculo de la carga en el hiperperiodo estacionario. Si el sistema tiene n trabajos en el hiperperiodo (n activaciones de sus tareas), el cálculo de obtener la carga al final del hiperperiodo I-ésimo será el de obtener la carga en el instante en que llega el trabajo de índice $j = nI$. Por tanto, para obtener experimentalmente los tiempos del método iterativo, lo que interesa es tener datos de coste en función del índice j.

Observar que el método iterativo parte siempre de una carga inicial cero, y por tanto $b = 1$, por lo que este parámetro tampoco juega ningún papel. No obstante, de cara a verificar experimentalmente la hipótesis de que el coste se incrementa linealmente con b, como se vio en la ecuación (6.1), se han realizado una serie de experimentos en los que se han analizado sistemas haciendo variar b mientras se dejaban fijos los restantes parámetros. Los resultados, mostrados en la figura 6.9 corroboran lo esperado. Observar que esta figura en particular no está en escala logarítmica.

Una vez establecido el comportamiento de b, en los restantes experimentos se hace $b = 1$, es decir, se parte de una carga inicial nula, como corresponde al método iterativo. Se han analizado miles de sistemas, con diferentes valores de $U^{\max}$ y m, y se ha medido el coste de obtener $\mathcal{W}(\lambda_j)$ para diferentes valores de j. Los resultados se muestran gráficamente (figuras 6.10 a 6.13), observándose que en escala logarítmica los costes se ajustan a una recta, lo que corrobora el crecimiento exponencial del coste. Se ha calculado la "recta" de regresión (en la escala logarítmica) de estos datos, que se representa también en las mismas figuras. De los parámetros de estas "rectas" de regresión, obtenemos los parámetros de la

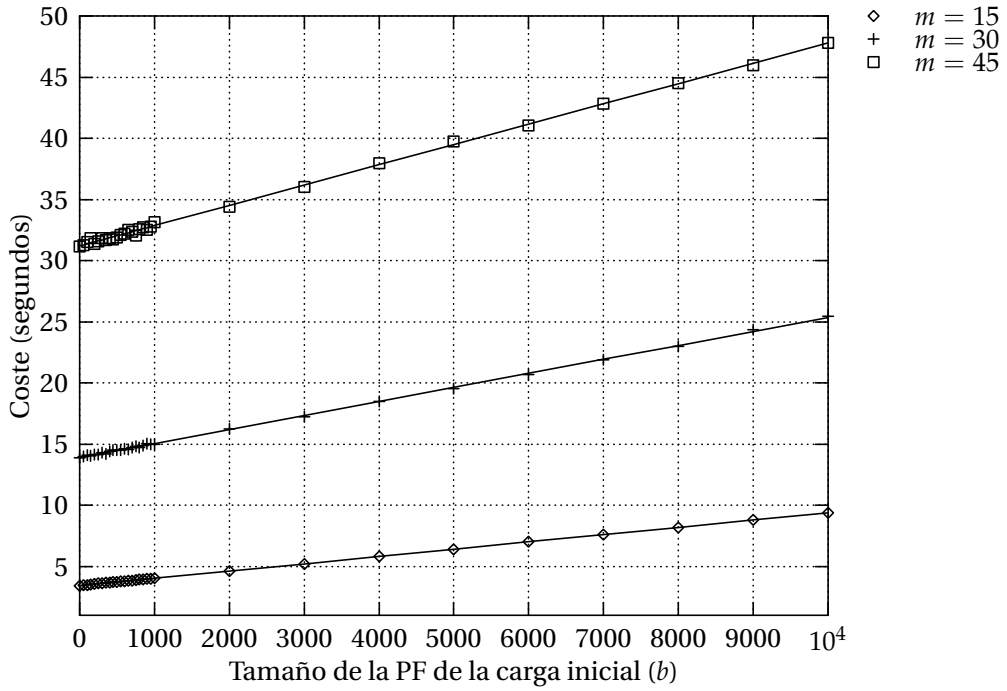


Figura 6.9.: Coste de obtener la carga del trabajo con índice $j = 1000$, en función de b y m, en sistemas en los que $U^{\max} \approx 5$

ecuación exponencial que gobierna el crecimiento del coste. Estas ecuaciones se muestran en el cuadro 6.1. En él podemos comprobar que el exponente no supera el valor 2 en ninguno de los casos, lo que concuerda con el crecimiento esperado de tipo $O(j^2)$ que se vio en la ecuación (6.2).

Cabe señalar que para la figura 6.10, correspondiente a $U^{\max} \approx 1$, el ajuste no es muy preciso, debido a que los datos experimentales presentan gran variabilidad. Para ese caso el coste computacional es tan bajo, que la precisión de la medida no es buena, pues se está en el límite de la resolución del cronómetro del sistema. Además, cualquier perturbación en el ordenador que está realizando el experimento puede causar una pequeñísima variación en su tiempo de cómputo, que es acusable a estas resoluciones tan bajas.

Coste del cálculo del tiempo de respuesta

Asumiendo que la PF de la carga $\mathcal{W}(\lambda_j)$ ya ha sido obtenida ¿cuál es el coste de obtener la PF de $\mathcal{R}_j$? En primer lugar hay que decir que depende del índice j, ya que cuanto mayor sea este índice, mayor será en general el número de puntos de $\mathcal{W}(\lambda_j)$ y por tanto mayor es el coste de las convoluciones siguientes. Además, también depende del parámetro k, que es el número promedio de interferencias que sufrirá el trabajo Γ_j, y que puede estimarse como la relación entre el plazo

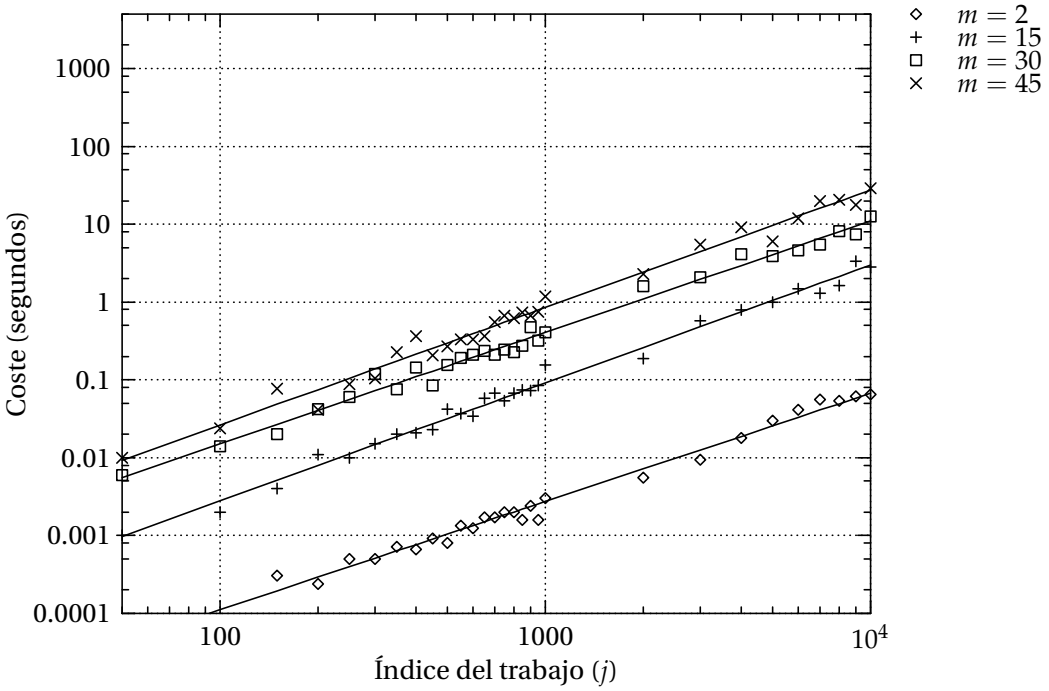


Figura 6.10.: Coste de obtener $\mathcal{W}(\lambda_j)$ en función de j, para diferentes valores de m en sistemas en los que $U^{\max} \approx 1$

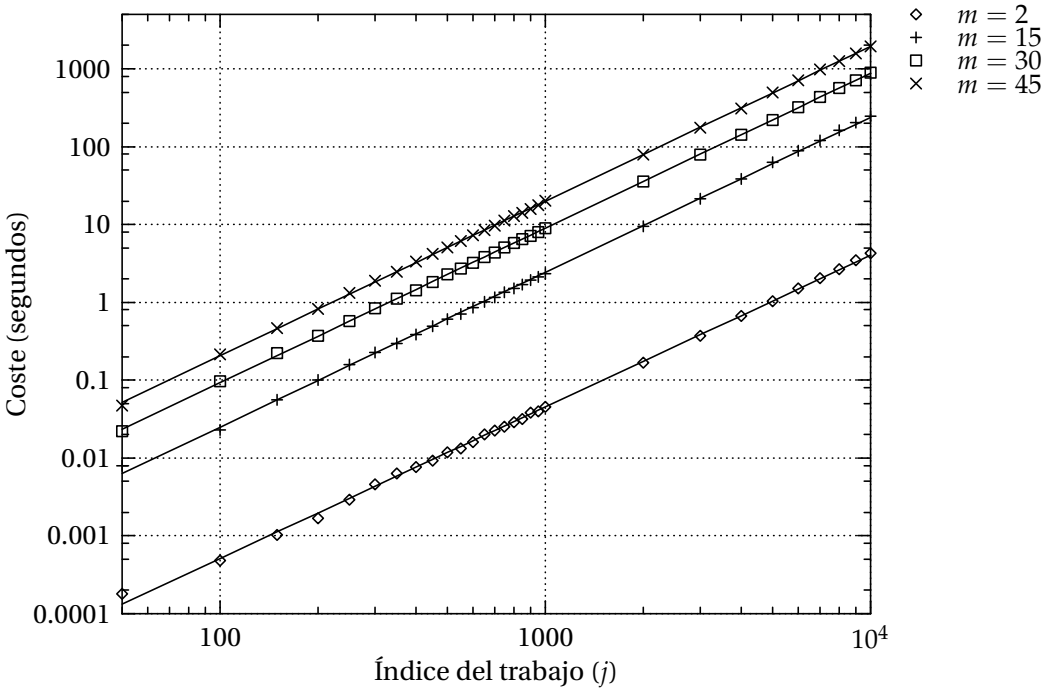


Figura 6.11.: Coste de obtener $\mathcal{W}(\lambda_j)$ en función de j, para diferentes valores de m en sistemas en los que $U^{\max} \approx 2$

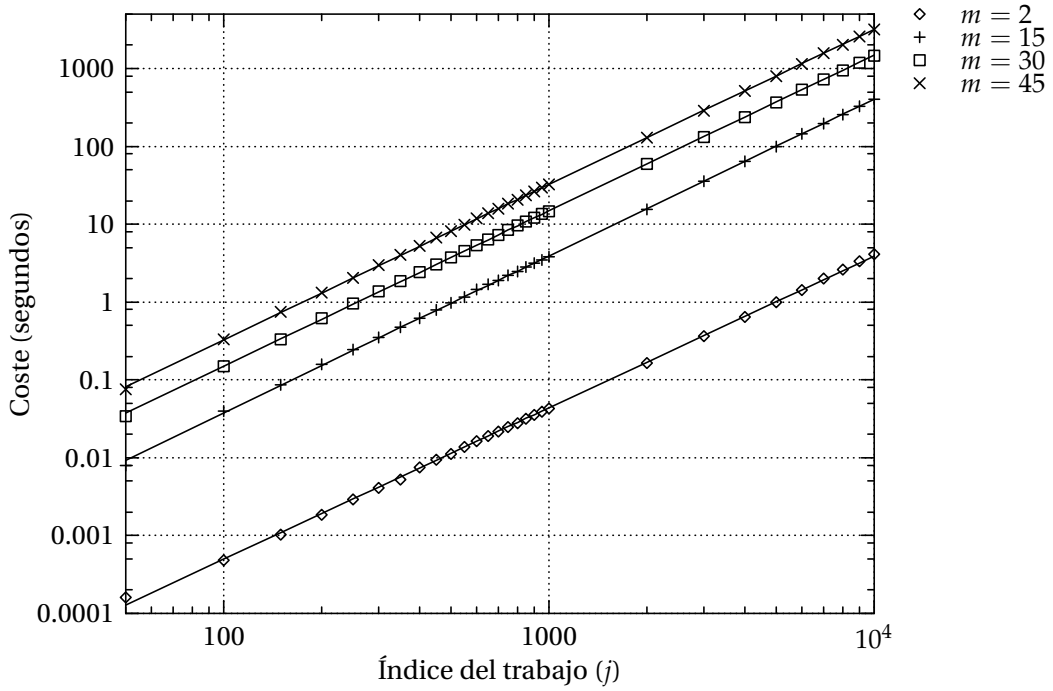


Figura 6.12.: Coste de obtener $\mathcal{W}(\lambda_j)$ en función de j, para diferentes valores de m en sistemas en los que $U^{\max} \approx 5$

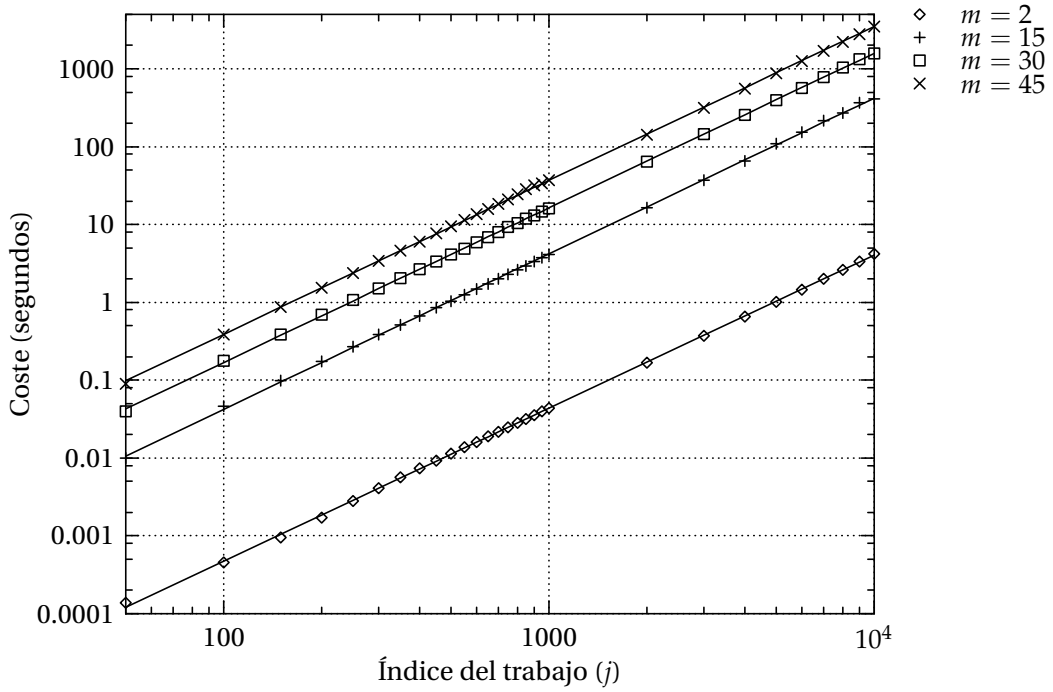


Figura 6.13.: Coste de obtener $\mathcal{W}(\lambda_j)$ en función de j, para diferentes valores de m en sistemas en los que $U^{\max} \approx 10$

	$U^{\max} \approx 1$	$U^{\max} \approx 2$	$U^{\max} \approx 5$	$U^{\max} \approx 10$
$m = 2$	$1{,}85\,10^{-7} \cdot j^{1{,}39}$	$6{,}55\,10^{-8} \cdot j^{1{,}95}$	$1{,}85\,10^{-7} \cdot j^{1{,}39}$	$5{,}58\,10^{-8} \cdot j^{1{,}96}$
$m = 15$	$2{,}58\,10^{-6} \cdot j^{1{,}52}$	$2{,}60\,10^{-6} \cdot j^{1{,}99}$	$2{,}58\,10^{-6} \cdot j^{1{,}52}$	$4{,}17\,10^{-6} \cdot j^{2{,}00}$
$m = 30$	$2{,}07\,10^{-5} \cdot j^{1{,}43}$	$9{,}78\,10^{-6} \cdot j^{1{,}98}$	$2{,}07\,10^{-5} \cdot j^{1{,}43}$	$1{,}80\,10^{-5} \cdot j^{1{,}99}$
$m = 45$	$2{,}52\,10^{-5} \cdot j^{1{,}51}$	$2{,}18\,10^{-5} \cdot j^{1{,}98}$	$2{,}52\,10^{-5} \cdot j^{1{,}51}$	$4{,}30\,10^{-5} \cdot j^{1{,}98}$

Cuadro 6.1.: Ecuaciones que ajustan a los datos experimentales de las figuras 6.10 a 6.13

promedio de todos los trabajos y el tiempo medio entre llegadas.

La gráfica de la figura 6.14 muestra la tendencia general del coste en función de k, para sistemas en los que $m = 20$. En esta gráfica se muestra cómo el coste de calcular $\mathcal{R}_j$, para un j dado, va aumentando dependiendo de k, o lo que es lo mismo, dependiendo del plazo de Γ_j. La gráfica muestra algunas curvas para diferentes valores de j. La utilización $U^{\max}$ también influye pues hace que los trabajos vengan más juntos o más separados y por tanto afecta al tamaño de la "cola" que es necesario convolucionar en el método "partir, convolucionar y juntar". En todo caso, por no complicar más las figuras, se ha representado sólo el caso en que la utilización máxima es grande ($U^{\max} \approx 10$). Ya se ha visto que incrementar más $U^{\max}$ produce poco incremento en el coste, pues tiende asintóticamente a un límite (recordar la figura 6.8).

Cabe señalar que la gráfica en escala logarítmica en este caso no es una línea recta. Esto se debe a que se combinan dos efectos en el coste computacional. De un lado, el parámetro k influye en el número de convoluciones que hay que realizar para obtener la PF del tiempo de respuesta, pero de otro lado el parámetro j influye indirectamente en el "tamaño" de estas convoluciones, ya que a medida que crece j, también crece el tamaño de la PF de $\mathcal{W}(\lambda_j)$, que es la que hay que convolucionar con $f_{\mathcal{C}_j}$ en la primera iteración del algoritmo.

Finalmente, se muestra (figuras 6.15 a 6.18) cómo aumenta el coste de obtener $\mathcal{R}_j$ en función de j, para diferentes valores de k. En cada gráfica k tiene un valor constante y se hace variar j. En todos los experimentos se ha hecho $\bar{T} \approx 1$, es decir, se consideran sistemas con alto factor de utilización máxima. De nuevo se observa que las gráficas no se ajustan bien a una línea recta, cuando k es grande. En realidad, se puede comprobar que el ajuste a una recta es bueno sólo a partir de un cierto valor de j. El valor concreto depende de k, así, en la figura 6.15 en la que k tiene un valor bajo, el ajuste a una recta se produce desde j muy bajo (de hecho, todos los puntos parecen ajustar bien a una recta). En cambio en la figura 6.18 en la que k es alto, el ajuste a la recta sólo se produce a partir de $j = 1000$. La explicación a esto radica de nuevo en que dos efectos independientes se conjugan

para dar el coste final. De un lado j influye en el tamaño de la carga con la que arranca el algoritmo, y de otro k influye en el número de veces que debe repetirse el algoritmo. Cuando k es bajo, el algoritmo se repite pocas veces y el coste está dominado por el tamaño inicial. Ya que este tamaño crece linealmente con j, el coste deberá crecer también linealmente (en efecto, la pendiente de las rectas en la figura 6.15 es próxima a 1). En cambio, cuando k es alto, y j es bajo, el número de veces que se repite el algoritmo tiene más importancia que el tamaño inicial de las funciones. Sólo cuando j comienza a ser del orden de k, la pendiente se aproxima a 1.

6.2.4. Predicción del coste de analizar un sistema dado con el método iterativo

Los resultados experimentales anteriores pueden usarse para predecir el coste de hallar el tiempo de respuesta de todos los trabajos para un sistema dado, asumiendo que se usa el método iterativo y que se sabe el número de iteraciones que este método requerirá para la convergencia.

Por ejemplo, supongamos un sistema formado por $N = 10$ tareas, cada una de ellas con un tiempo de ejecución definido por $m = 30$ puntos, y con un factor de utilización máximo $U^{\max} = 10$ (la utilización promedio no es importante). Los plazos son iguales a los periodos para todas las tareas.

Para poder usar los datos y fórmulas de la sección anterior, necesitamos convertir esta información en los parámetros m, n y k que se representan en las gráficas. El dato m viene directamente del modelo de tareas, y es $m = 30$, pero los restantes deben ser calculados en función de los periodos de las tareas. Así tenemos que:

$$n = \sum_{i=1}^{N} \frac{T}{T_i}$$

siendo T la longitud del hiperperiodo, es decir, el mínimo común múltiplo de los T_i. Para calcular n, necesitamos por tanto los periodos de las tareas del sistema. Supongamos que en nuestro ejemplo éstas son tales que producen $n = 500$.

En cuanto a k, tal como ha sido definida es el "número promedio de expulsiones" sufridas por los trabajos, en un intervalo de longitud igual a su plazo. Ya hemos visto que:

$$k = \frac{1}{n} \sum_{j=1}^{n} k_j = \frac{1}{n} \sum_{j=1}^{n} \frac{D_j}{\bar{T}}$$

Ya que $\bar{T}$ no depende de j, puede salir del sumatorio:

$$k = \frac{1}{n\bar{T}} \sum_{j=1}^{n} D_j$$

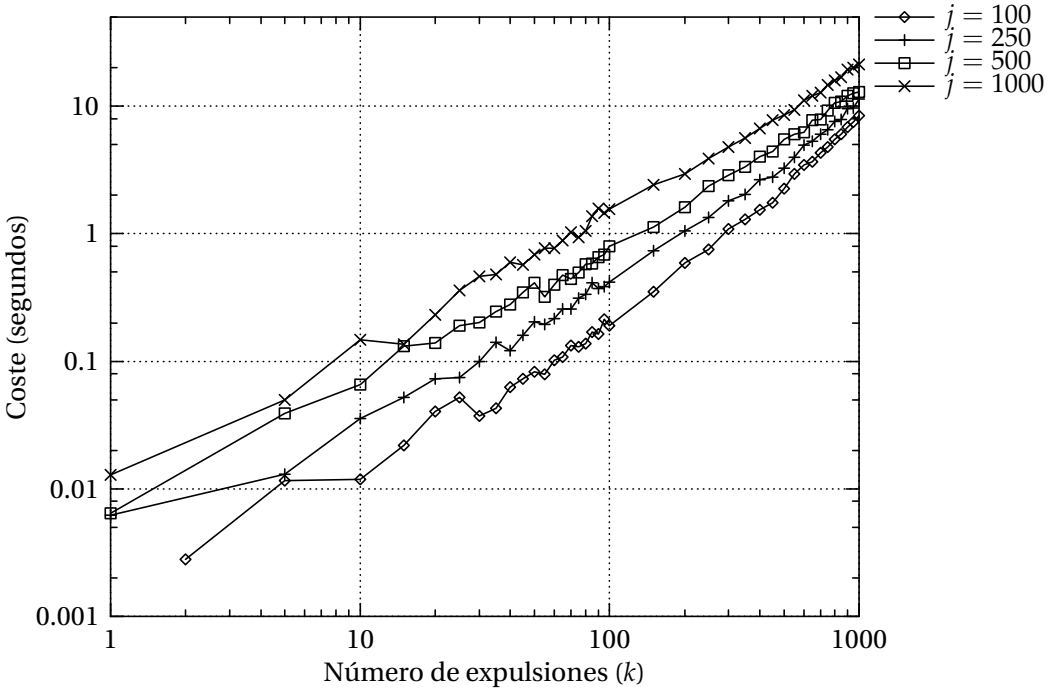


Figura 6.14.: Coste de obtener $\mathcal{R}j$ en función de k, para diferentes valores de j en sistemas en los que $U^{\max} \approx 10, m = 20$

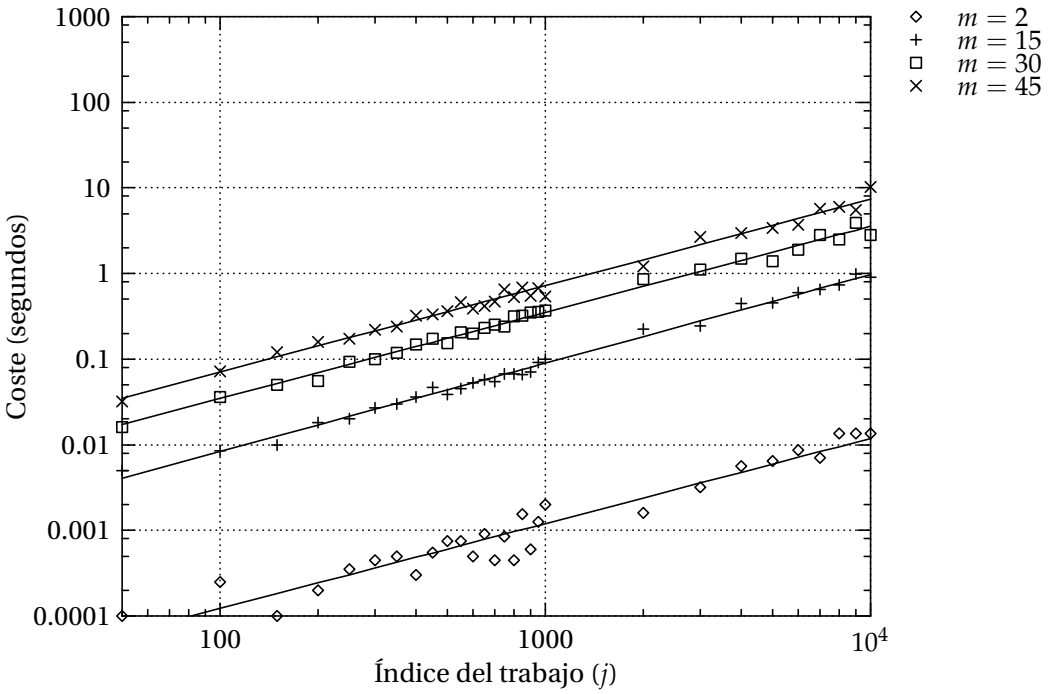


Figura 6.15.: Coste de obtener $\mathcal{R}_j$ en función de j y m, en sistemas con alta utilización y $k = 10$

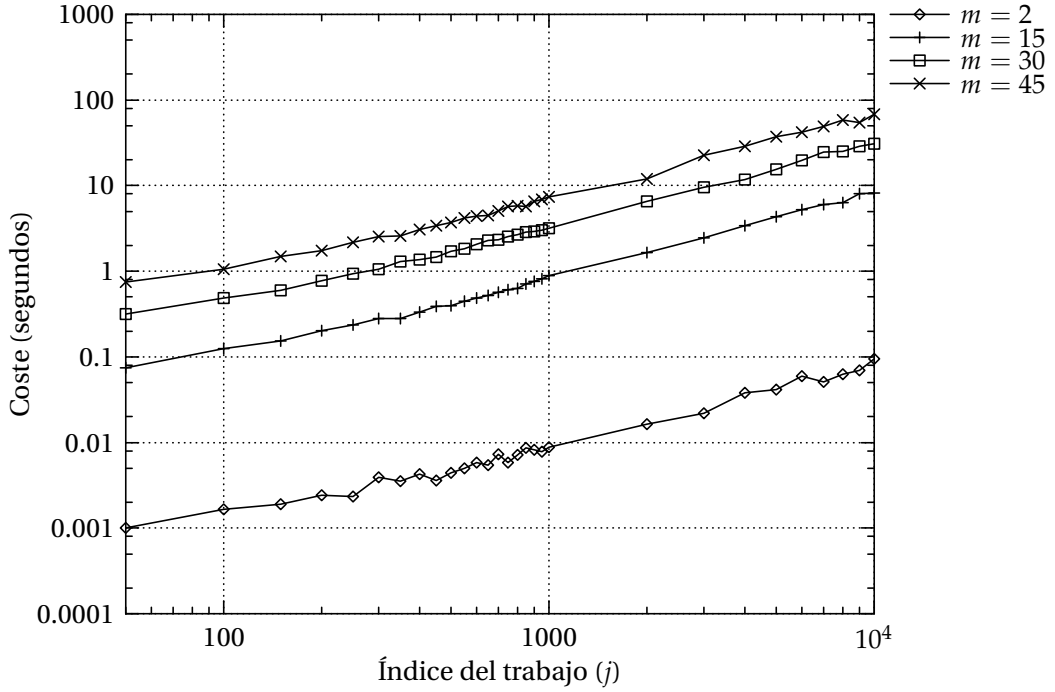


Figura 6.16.: Coste de obtener $\mathcal{R}_j$ en función de j y m, en sistemas con alta utilización y $k = 100$

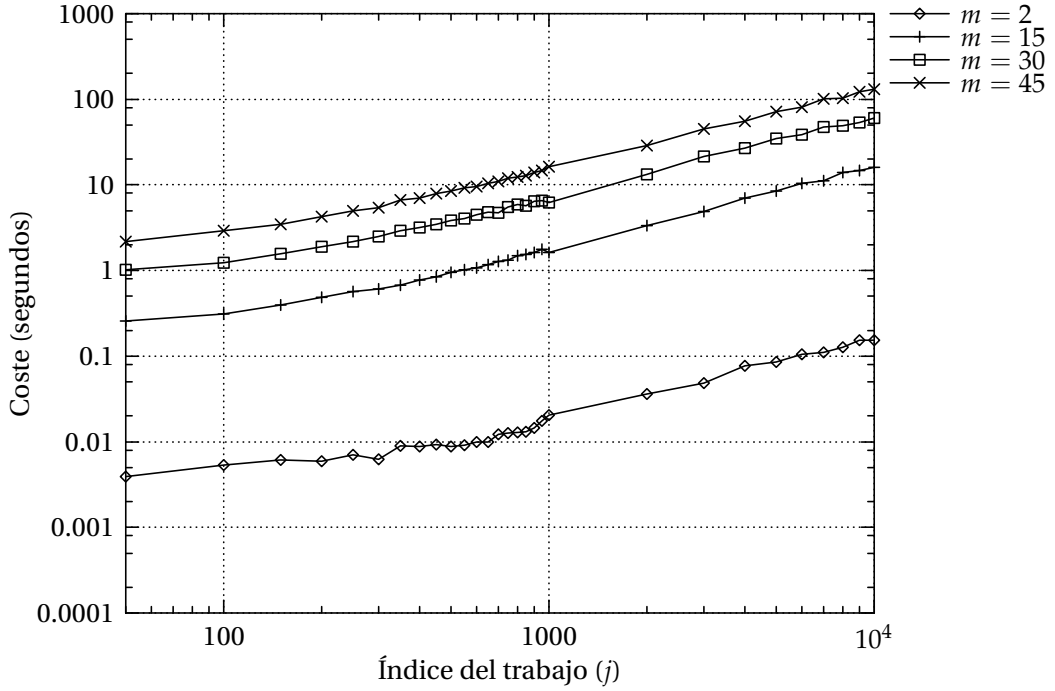


Figura 6.17.: Coste de obtener $\mathcal{R}_j$ en función de j y m, en sistemas con alta utilización y $k = 200$

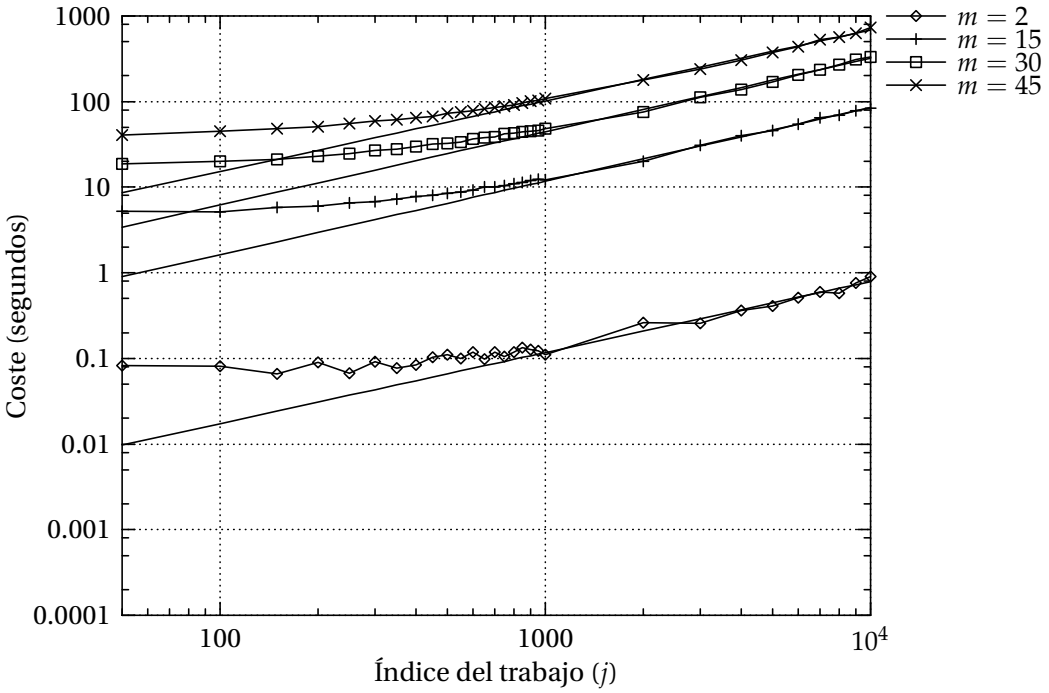


Figura 6.18.: Coste de obtener $\Re_j$ en función de j y m, en sistemas con alta utilización y $k = 1000$

El sumatorio incluye los plazos de todos los *trabajos* activados en un hiperperiodo. Si queremos ponerlo en función de las *tareas*, hay que tener en cuenta que cada tarea τ_i es activada T/T_i veces por hiperperiodo, y que todas esas activaciones tienen el mismo plazo relativo. Por tanto la expresión anterior puede ponerse en función de las N tareas, en lugar de los n trabajos:

$$k = \frac{1}{n\bar{T}} \sum_{i=1}^{N} D_i \frac{T}{T_i}$$

Finalmente, ya que $n\bar{T} = T$, éste se simplifica con el T que hay en el interior del sumatorio y llegamos finalmente a la ecuación que nos permite obtener k a partir de los parámetros del sistema de tareas periódicas:

$$k = \sum_{i=1}^{N} \frac{D_i}{T_i} \tag{6.15}$$

En nuestro ejemplo, al ser los plazos iguales a los periodos, cada término del sumatorio es 1, por lo que finalmente $k = N$, es decir, k coincide con el número de tareas que es 10 en este ejemplo.

Una vez tenemos todos los parámetros, podemos usar los datos experimentales en la forma siguiente.

1. Determinar el coste de hallar la carga para el hiperperiodo estacionario por el método iterativo.

   Supongamos que se requieren 5 iteraciones de este método. Ya que cada hiperperiodo se compone de $n = 100$ trabajos, la carga al final del quinto hiperperiodo es la carga en el instante de llegada del trabajo $j = 500$. De modo que el coste que buscamos es el coste del cómputo de la carga para $j = 500$, $m = 30$, $U^{\max} = 10$. Este coste se obtiene de la gráfica 6.11, o si se prefiere aplicando las fórmulas del cuadro 6.1 (en este caso se usaría la fórmula sombreada en gris).

   El resultado es $c_{\mathcal{W}_{500}} \approx 4{,}13$ segundos.

2. Determinar el coste de hallar la carga para cada uno de los $n = 100$ trabajos que se activan en el "hiperperiodo estacionario".

   En realidad se trata del coste de hallar la carga para los trabajos que se activan en el quinto hiperperiodo (en el que suponemos se ha alcanzado la estacionariedad), es decir, para los trabajos con índice $j = 501, 502, \ldots, 599$, pero debe tenerse en cuenta que para calcular la carga de cada uno de ellos no se recomienza de nuevo desde $j = 0$, sino que se puede reaprovechar el resultado del un trabajo anterior (trabajo base).

   En un caso muy pesimista (prácticamente imposible) el único trabajo basal del hiperperiodo sería el primero, por lo que para calcular la carga de los trabajos $j = 501, \ldots, 599$ habría que recomenzar siempre desde el trabajo $j = 500$. En este caso, el coste de obtener la carga del trabajo j sería $c_{\mathcal{W}_j} - c_{\mathcal{W}_{500}}$. El coste de obtener la carga para los $n = 100$ trabajos del hiperperiodo será por tanto la suma de lo que cuesta el primero $c_{\mathcal{W}_{500}}$ más el sumatorio de lo que cuesta cada uno de los restantes. Por tanto, el coste total será

   $$\text{Coste} = c_{\mathcal{W}_{500}} + \sum_{j=501}^{599} \left(c_{\mathcal{W}_j} - c_{\mathcal{W}_{500}} \right)$$

   Evaluando este sumatorio con ayuda de la fórmula del cuadro 6.1 se obtiene un coste de 87 segundos. Para evitar tener que evaluar el sumatorio se puede hacer una aproximación aún más pesimista. Teniendo en cuenta que los costes son crecientes con j, se tiene que para cualquier término del sumatorio $c_{\mathcal{W}_{600}} \geq c_{\mathcal{W}_j}$ y por tanto

   $$\text{Coste} \leq c_{\mathcal{W}_{500}} + 99(c_{\mathcal{W}_{600}} - c_{\mathcal{W}_{500}})$$

   Aplicando esta fórmula se obtiene un coste (muy pesimista) de unos 180 segundos, o sea, tres minutos.

Pero la suposición de que el primer trabajo del hiperperiodo es el único trabajo basal es demasiado pesimista. En EDF es común que la mayor parte de los trabajos sean basales. En RM y DM todos los trabajos (correspondientes a la tarea de menor prioridad) son basales. Si asumimos un caso así, el resultado de cada iteración servirá para la siguiente, no siendo necesario "retroceder y rehacer" nunca. En este caso el coste de obtener la carga para los 100 trabajos del hiperperiodo coincide con el de obtenerla para el último de ellos (ya que para llegar al último hay que calcular la de los anteriores, que se puede ir almacenando). Por tanto, en ese caso el coste sería $c_{\mathcal{W}_{600}}$, que resulta ser de unos 6 segundos.

Sabemos pues que el coste computacional de encontrar la carga en los instantes de activación de todos los trabajos se mueve entre unos 6 segundos (caso más favorable, todos los trabajos son basales) y 87 segundos (más desfavorable, todos los trabajos son no-basales, y además para todos ellos el único trabajo base es el primero del hiperperiodo).

3. Determinar el coste de obtener los tiempos de respuesta de todos los trabajos en el hiperperiodo estacionario.

   Para conocer el coste de obtener el tiempo de respuesta de un único trabajo de índice j, se utilizarán los datos de las gráficas 6.15 a 6.18. Así, por ejemplo, para obtener el coste para el primer trabajo del hiperperiodo estacionario (que hemos asumido que comienza en el trabajo $j = 500$), acudiríamos a la figura 6.15 (puesto que $k = 10$ en este sistema, como hemos visto), con $j = 500$ y $m = 30$, y obtenemos un coste de unos 0,15 segundos.

   Para conocer el tiempo total de análisis de los $n = 100$ trabajos, podemos consultar el coste de cada uno de ellos en la misma figura, entrando sucesivamente con los valores $j = 501, 502, \ldots, 599$ y finalmente sumarlos todos, o podemos simplificar (hacia el lado del pesimismo) y mirar únicamente el coste del último de ellos ($j = 599$) que será el mayor de todos, y multiplicar éste por $n = 100$. Haciéndolo de esta segunda forma el coste obtenido es $0{,}2 \times 100 = 20$ segundos.

4. El coste total del análisis será la suma de los tres tiempos anteriores, por tanto estará comprendido entre los 30 segundos (si todos los trabajos son basales) y los 2 minutos (si todos los trabajos son no-basales).

Observar que en la práctica el coste será bastante menor que el obtenido de la estimación anterior, ya que los datos de las gráficas son para el caso en que todos los trabajos del sistema tengan mayor prioridad que el que se está analizando.

En la práctica, cuando se analiza un trabajo de prioridad intermedia, no es necesario considerar la interferencia ni la carga originada por los trabajos de menor prioridad, por lo que el número de convoluciones disminuye. Si se quisiera una

estimación más precisa, usando las mismas gráficas, habría que interpretar j no como "el índice del trabajo que se está analizando", sino como "el número de trabajos de mayor o igual prioridad que llegan antes del trabajo analizado", y k como "el número de trabajos de mayor prioridad que llegan después del trabajo analizado, y antes de su plazo absoluto". En este sentido, para un sistema planificado con prioridades fijas, el análisis de cada nivel de prioridad tiene coste diferente, haciéndose cada vez menor cuanto mayor es la prioridad del nivel analizado.

6.3. Comparación con otros métodos

De los restantes métodos de análisis estocástico que se han tratado en el capítulo sobre "trabajo relacionado" (capítulo 2), no todos se pueden comparar con el propuesto en esta tesis, ya que algunos requieren un planificador específico (como los métodos de servidor de ancho de banda constante de Abeni y Buttazzo [2001]); otros plantean un modelo de sistema bastante diferente que incluye dependencias entre tareas pero prohíbe la expulsión de las mismas o los plazos mayores que los periodos (como los trabajos de Manolache [2002] o Kalavade y Moghê [1998]).

De hecho, los pocos métodos que son comparables al presentado en esta tesis son el de Tia *et al.* [1995] y el de Gardner [1999]. Ambos plantean un modelo prácticamente idéntico al de esta tesis (aunque no abordan ninguna de las extensiones del capítulo 5), pero lo resuelven de forma diferente, mediante simplificaciones que causan pesimismo en la estimación de las probabilidades.

Compararemos estos dos enfoques entre sí y con el propuesto en esta tesis mediante unos sencillos casos de ejemplo, extraídos de [Gardner, 1999]. Analizaremos el mismo conjunto de ejemplos mediante cuatro métodos:

1. La aproximación de Tia *et al.* [1995], que denominaremos **PTDA**[4], siguiendo a Gardner [1999]. Esta aproximación no encuentra la PF del tiempo de respuesta, sino la probabilidad de cumplir los plazos (de la que se deduce la probabilidad de perderlos). La hipótesis simplificadora es que los plazos han de ser iguales a los periodos y la utilización máxima menor que uno, y se centra en analizar la carga que se genera en un instante crítico, y calcular la probabilidad de que esta carga haya sido servida en su totalidad antes de la llegada del siguiente trabajo.

2. La aproximación de Gardner [1999], que él mismo denomina **STDA**[5]. Esta aproximación sí encuentra la PF del tiempo de ejecución, pero sólo para la tarea "peor" dentro de un periodo ocupado que comience en un instante

[4] *Probabilistic Time Demand Analysis*
[5] *Stochastic Time Demand Analysis*

crítico. De nuevo hace la suposición de que los plazos han de ser menores o iguales que los periodos.

3. La solución exacta que se obtiene aplicando el **análisis estocástico** propuesto en **esta tesis**. Esta solución es la PF del tiempo de respuesta de cada trabajo, de la cual se puede obtener la probabilidad de pérdida de plazo de cada trabajo y por tanto de cada tarea.

4. Los resultados de **simular** un sistema sintético que tenga los mismos parámetros que el que se quiere resolver. En la simulación podemos contar simplemente el número de plazos que se pierden, de donde podemos obtener la frecuencia de pérdidas de plazo que será igual a la probabilidad de pérdida de plazo si la simulación dura un tiempo suficientemente largo. Pero también podemos medir en la simulación cuál es el tiempo de respuesta de cada trabajo, y construir así un "histograma" de tiempos que coincidirá con la PF del tiempo de respuesta, si la simulación dura lo suficiente.

6.3.1. Sistemas de ejemplo a resolver

Resolveremos tres sencillos sistemas que llamaremos S_1, S_2 y S_3, cuyos parámetros se resumen en el cuadro 6.2. Estos ejemplos están copiados directamente de la página 43 de [Gardner, 1999] (si bien allí los ejemplos se numeran 4, 5 y 6 en lugar de 1, 2 y 3).

Estos tres sistemas tienen la particularidad de presentar la misma utilización promedio $\bar{U}$, debido a que el tiempo de ejecución promedio de las tareas es el mismo en los tres, así como sus periodos y plazos. Lo que varía son los tiempos de ejecución máximo y mínimo de cada tarea, y por consiguiente la utilización máxima y mínima de cada sistema. Los valores de estos parámetros se muestran también en el cuadro 6.2. Se asume que la distribución de los tiempos de ejecución es uniforme.

		Parámetros						Utilización		
Sistema	Tarea	T_i	Φ_i	D_i	$C_i^{\min}$	$\bar{C}_i$	$C_i^{\max}$	$U^{\min}$	$\bar{U}$	$U^{\max}$
S_1	τ_1	300	0	300	72	100	128	0,4200	0,7083	0,9967
	τ_2	400	0	400	72	150	228			
S_2	τ_1	300	0	300	50	100	150	0,2917	0,7083	1,1250
	τ_2	400	0	400	50	150	250			
S_3	τ_1	300	0	300	1	100	199	0,0058	0,7083	1,4108
	τ_2	400	0	400	1	150	299			

Cuadro 6.2.: Parámetros de los sistemas a resolver, tomados de [Gardner, 1999]

Como se observa, los tres sistemas tienen la misma duración del hiperperiodo $T = 1200$. La tarea τ_1 de mayor prioridad se activa cuatro veces en el hiperperiodo, en los instantes $0, 300, 600$ y 900. Puesto que es la de máxima prioridad, no sufre retraso ni interferencia, y cumplirá siempre sus plazos, por lo que no realizaremos análisis sobre ella, centrándonos únicamente en la probabilidad de pérdida de plazo para la tarea τ_2.

El patrón de activaciones, que es el mismo para los tres sistemas, se muestra en la figura 6.19.

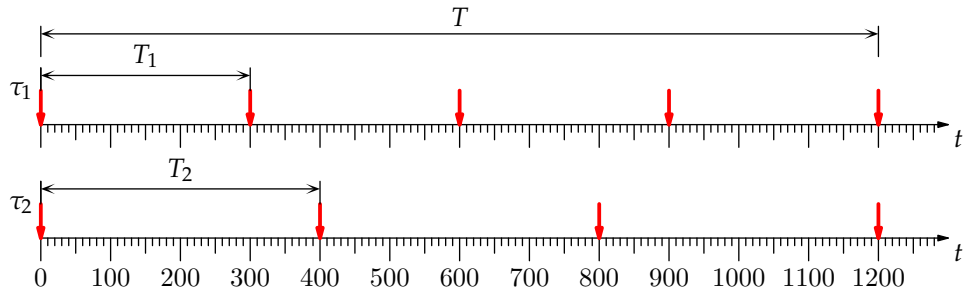


Figura 6.19.: Patrón de activaciones de tareas para los tres sistemas de ejemplo

6.3.2. Resultados

Se calcula la probabilidad de pérdida de plazo para la tarea τ_2 utilizando los cuatro métodos antes detallados. El cuadro 6.3 compara los resultados obtenidos por cada uno de los métodos. Cabe destacar que las aproximaciones de Tia y de Gardner son muy pesimistas, mientras que los resultados de nuestro análisis son exactos, como se observa comparando con los resultados obtenidos por simulación. Seguidamente entraremos en más detalles acerca de cómo se obtuvo cada entrada en dicho cuadro.

	Sistema		
Método	S_1	S_2	S_3
PTDA [Tia *et al.*, 1995]	0,186303	0,259535	0,331104
STDA [Gardner, 1999]	0,140100	0,489000	0,608000
Análisis estocástico (esta Tesis)	0,047058	0,073572	0,192204
Simulación [Gardner, 1999]	$0{,}047 \pm 0{,}1$	$0{,}074 \pm 0{,}2$	$0{,}192 \pm 0{,}1$

Cuadro 6.3.: Probabilidad de que τ_2 pierda su plazo, según los diferentes métodos

Obtención de los resultados

Aproximación PTDA Esta aproximación parte de un instante crítico en que llegan todas las tareas del sistema a la vez, y se asume el origen de tiempos en ese instante. La carga generada en el intervalo $[0, t)$ se calcula como la suma de los tiempos de ejecución de los trabajos que lleguen en ese intervalo. Esto se convierte en la convolución de las PF de sus tiempos de ejecución. Una vez obtenida esta PF, se puede calcular la probabilidad de que esta carga sea menor o igual que t, lo cual indicaría que toda la carga generada se ha servido y que por tanto el "periodo ocupado" ha terminado. Esto nos da la probabilidad de que la tarea de menor prioridad finalice antes de t.

El método va repitiendo el procedimiento anterior para diferentes t, hasta alcanzar un t igual al plazo absoluto de la tarea bajo consideración. Para cada t considerado se obtiene una probabilidad de que la tarea finalice antes de t. Finalmente, se queda con la probabilidad más baja de todas las encontradas. La probabilidad de que pierda plazo será 1 menos la hallada. Para evitar el tener que considerar todos los posibles t antes del plazo, el método propone utilizar sólo como "instantes t" los instantes de activación de nuevos trabajos.

Apliquemos pues estas ideas al cálculo de la probabilidad de perder el plazo de la tarea τ_2 para los sistemas de ejemplo del cuadro 6.2. El instante crítico es $t = 0$, en el que llegan τ_1 y τ_2. La secuencia de "instantes t" en los que se debe evaluar la probabilidad son todas las llegadas de nuevos trabajos en el intervalo $[0, D_2]$. Ya que $D_2 = 400$, los únicos instantes a considerar son $t = 300$ y $t = 400$, como se observa en la figura 6.19.

En el intervalo $[0, 300)$ la carga generada es $\mathcal{C}_1 + \mathcal{C}_2$. Convolucionando la PF de ambas funciones obtenemos una función $f_{\mathcal{C}_1+\mathcal{C}_2}(\cdot)$, de la que podemos obtener la probabilidad $\mathbb{P}\{\mathcal{C}_1 + \mathcal{C}_2 \leq 300\}$, que constituye una primera estimación de la probabilidad de finalizar antes del plazo. Por ejemplo, para el sistema S_1 se obtiene 0,821656

En el intervalo $[0, 400)$ la carga generada es $\mathcal{C}_1 + \mathcal{C}_2 + \mathcal{C}_1$, puesto que la tarea τ_1 se activa dos veces. Convolucionando estas tres funciones se obtiene $f_{\mathcal{C}_1+\mathcal{C}_2+\mathcal{C}_1}(\cdot)$ de donde se puede obtener $\mathbb{P}\{\mathcal{C}_1 + \mathcal{C}_2 + \mathcal{C}_1 \leq 400\}$, lo que constituye otra estimación de la probabilidad de finalizar antes del plazo. Por ejemplo, para el sistema S_1 se obtiene 0,813697.

Finalmente se toma como resultado del análisis la menor de las dos probabilidades antes calculadas. Así, para el ejemplo S_1 se obtiene 0,813697, que se considera un límite inferior a la probabilidad de cumplir el plazo. Por tanto 1 menos esa cantidad será un límite superior a la probabilidad de perderlo. El resultado es 0,186303. Como se observa en el cuadro 6.3 es una estimación bastante pesimista, puesto que la probabilidad real, corroborada por simulación, es del orden de 0,047.

De forma análoga se calcula para los sistemas S_2 y S_3. Aunque, en rigor, el mé-

todo PTDA no puede usarse en sistemas con $U^{\max} > 1$, resulta que los resultados obtenidos para este caso también son válidos. De hecho, si los comparamos con los de Gardner [1999], son incluso más ajustados. No obstante, en general, para utilizaciones $U^{\max}$ más altas, la aproximación PTDA deja de proporcionar un límite de probabilidad válido, debido a que considera que en el instante crítico $t = 0$ la carga es nula, cuando sabemos que hay una probabilidad de que exista carga pendiente que proviene del hiperperiodo anterior.

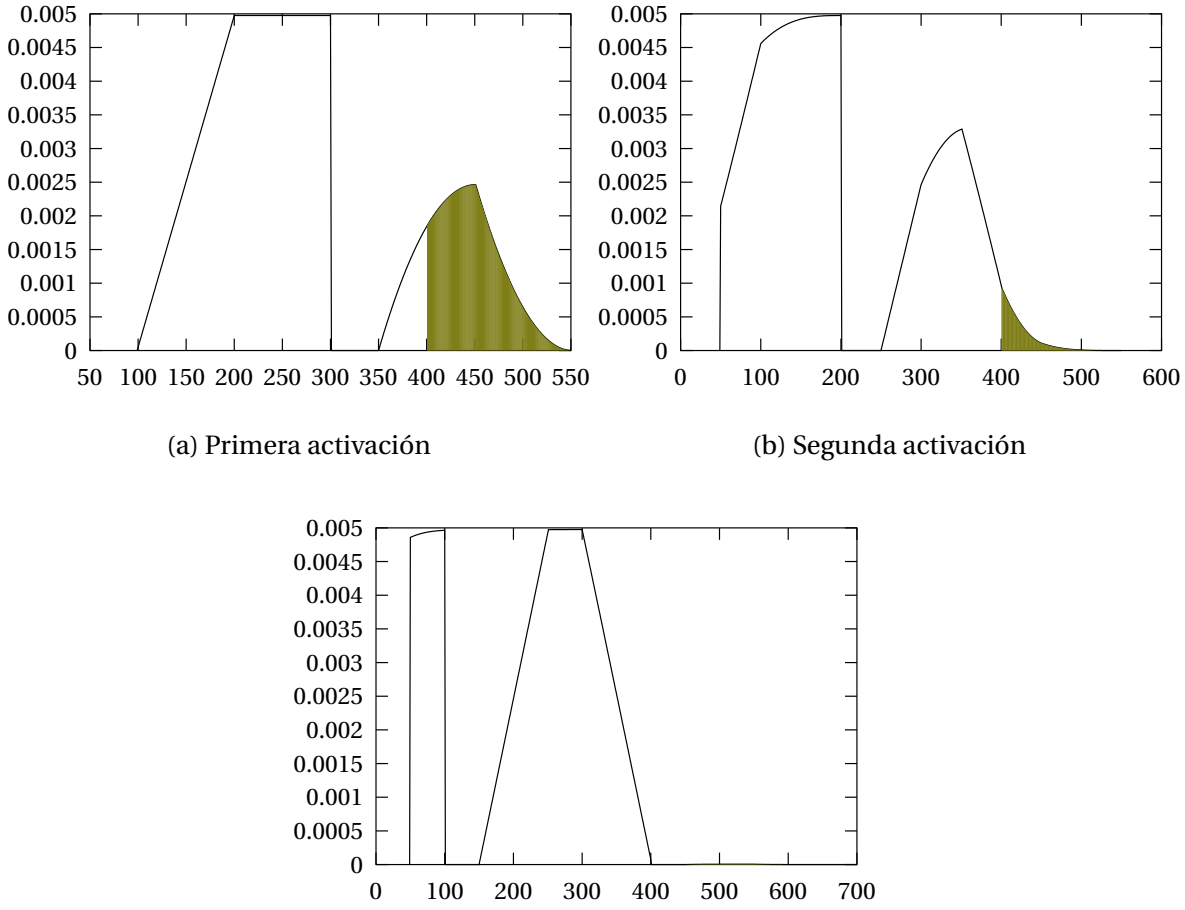
Aproximación STDA El método de Gardner [1999] comienza igual que el PTDA, asumiendo un instante crítico en el que todas las tareas se activan a la vez. Calcula la PF del tiempo de respuesta de la tarea de menor prioridad para este caso (usando una técnica similar al "partir, convolucionar y juntar"), y a partir de ella, la probabilidad de que esta primera activación pierda su plazo.

A partir de esta misma distribución se puede obtener también la probabilidad de que la tarea aún esté en ejecución cuando llegue su próxima instancia (de hecho, al ser los plazos iguales a los periodos, esta probabilidad coincide con la hallada antes). Cuando esta probabilidad es mayor de 0, es necesario analizar también el tiempo de respuesta de la siguiente activación, ya que puede ser mayor que el de la primera. Y así sucesivamente, se proseguiría hasta que una de las tareas termine con probabilidad 1 antes de que llegue su siguiente activación. Observar que esto ocurrirá sólo cuando $U^{\max} < 1$. En otro caso será necesario analizar infinitas tareas, si bien las probabilidades de pérdida de plazo irán convergiendo (Gardner no demuestra esto, pero nosotros sí hemos demostrado que en efecto es así siempre que $\bar{U} < 1$). Finalmente, se toma la mayor probabilidad de pérdida de plazo de entre todas las activaciones analizadas.

Sin embargo parece haber algo erróneo en el método usado por Gardner para calcular la PF del tiempo de respuesta para activaciones distintas de la primera. En teoría, debería considerar de alguna forma la PF de la carga en el instante de la activación, pero sin embargo no se detalla claramente el método usado. Aparentemente, lo que Gardner hace es considerar que la PF de la carga es igual a la "cola" de la PF del tiempo de respuesta de la activación anterior, cortada en el instante en que llega la activación actual, y renormalizada para que su suma sea 1. Esto es incorrecto y podría ser la causa de los pésimos resultados que se observan en el cuadro 6.3, para los sistemas S_2 y S_3.

Por ejemplo, para el sistema S_2, la PF del tiempo de respuesta de la primera activación de τ_2 es la representada en la figura 6.20(a), de la que se puede obtener la probabilidad de pérdida de plazo, que sale 0,19673 (área sombreada). La de la segunda activación, obtenida con el método expuesto en esta tesis, debería obtenerse calculando antes la PF de la carga en $t = 400$. Haciéndolo de este modo se obtendría la mostrada en la figura 6.20(b), que tiene una probabilidad de perder su plazo de 0,02342. Finalmente, la PF del tiempo de respuesta de la terce-

ra activación, obtenida por los métodos descritos en esta tesis, sería la mostrada en la figura 6.20(c), que tiene una probabilidad de pérdida de plazo de 0,00051. Siguiendo a Gardner nos quedaríamos ahora con la mayor de estas tres, que es 0,19673. Esto no coincide con el valor 0,489 mostrado en el cuadro 6.3, que es el que Gardner consigna en su trabajo [Gardner, 1999, página 43].



(a) Primera activación

(b) Segunda activación

(c) Tercera activación

Figura 6.20.: Funciones de probabilidad del tiempo de respuesta de las tres primeras activaciones de τ_2 en el sistema S_2

De igual modo, la solución que Gardner obtiene para el sistema S_3 (0,608) tampoco coincide con la que se obtiene tomando la menor de las tres primeras activaciones, que sería 0,261986. Mi explicación es que Gardner comete algún error al computar la PF de la carga en instantes distintos de $t = 0$, por lo que el tiempo de respuesta de la primera activación sale correcto, pero el de las restantes activaciones sale "peor" de lo que debería.

Análisis estocástico propuesto en esta tesis La tarea τ_2, de menor prioridad, se activa tres veces en el hiperperiodo, en los instantes 0, 400 y 800. El análisis exacto propuesto en esta tesis obtiene la PF del tiempo de respuesta para cada uno de estos tres casos y promediando los tres se obtiene la PF del tiempo de respuesta de la tarea. A partir de esta PF es posible hallar la probabilidad de pérdida de plazo de forma exacta.

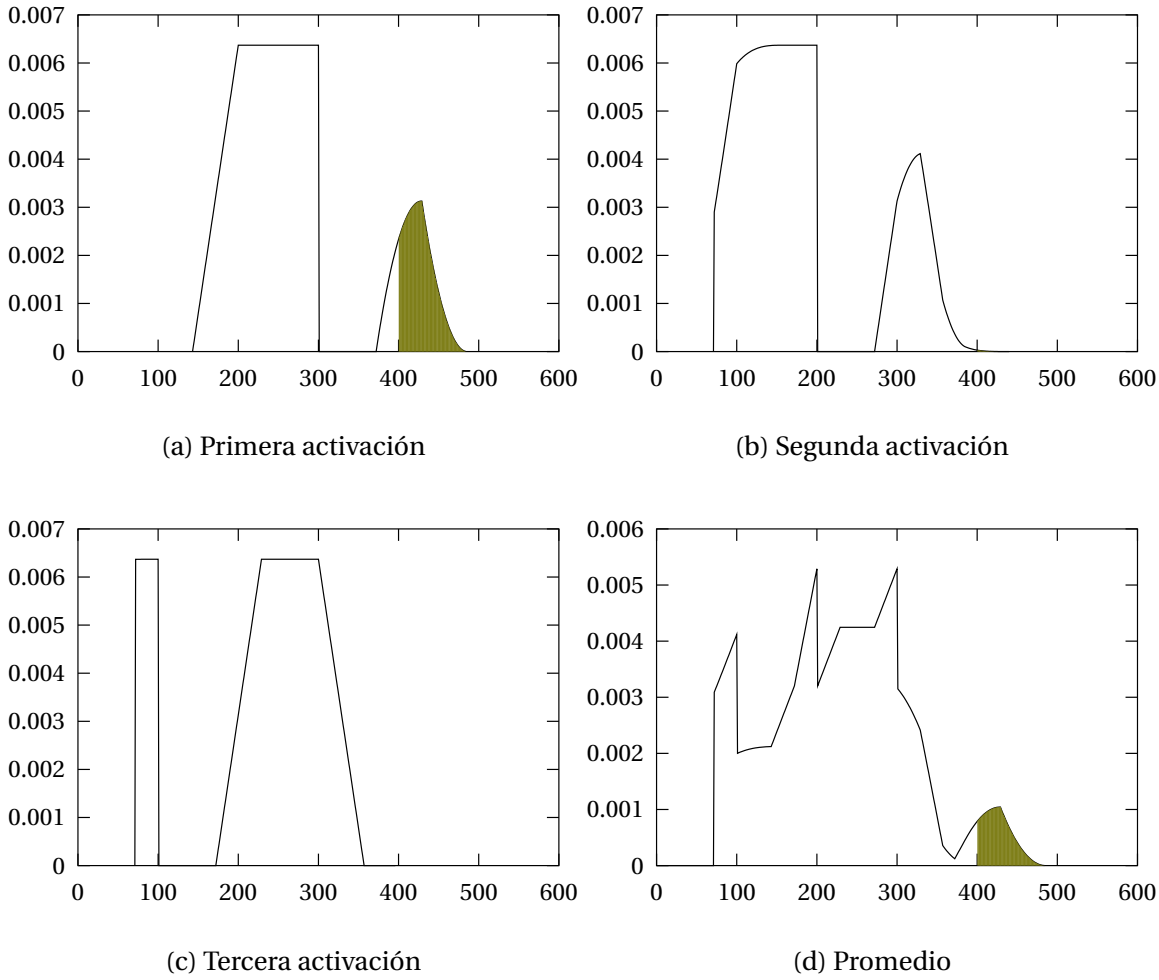
Para obtener estas PFs, es necesario un análisis Markoviano en los casos en los que $U^{\max} > 1$, es decir, en los sistemas S_2 y S_3. Sin embargo, para el caso S_1 no es necesario tal análisis sino que se puede obtener la probabilidad exacta analizando únicamente el primer hiperperiodo.

Las gráficas de los tiempos de respuesta para cada una de las tareas, así como la gráfica promedio de todas las activaciones se muestran en las figuras 6.21, 6.22 y 6.23, para los sistemas S_1, S_2 y S_3, respectivamente. En los dos últimos casos, la gráfica es mucho más larga (en realidad es infinita, puesto que existe una probabilidad no nula, aunque pequeñísima, de que el tiempo de respuesta alcance cualquier valor arbitrario), pero ha sido truncada a sus primeros puntos para compararla más fácilmente con las del caso S_1. El tiempo requerido por la herramienta para hallar estas distribuciones ha sido de unas milésimas de segundo para el sistema S_1, y apenas medio segundo para S_2 y S_3 (incluyendo el análisis Markoviano por iteración hasta una precisión de 10^{-12}). En estas figuras el área sombreada es la probabilidad de pérdida de plazo.

Simulación Los resultados correspondientes a los experimentos de simulación consignados en la tabla 6.3 han sido copiados de [Gardner, 1999, página 43]. En este trabajo el autor menciona que el tiempo de simulación necesario para alcanzar esta precisión fue de 10 minutos.

Los experimentos han sido repetidos con otro simulador programado por mí, y se ha llegado a los mismos resultados. Adicionalmente mi simulador almacena el tiempo de respuesta que ha tenido cada activación de la tarea y construye un histograma con estos tiempos. Si normalizamos el histograma de forma que su integral sea 1, debería coincidir con la función de probabilidad hallada por el análisis exacto expuesto en el apartado anterior.

En la figura 6.24 se muestran los histogramas obtenidos de la simulación para cada uno de los sistemas y, superpuesta a los mismos, la curva teórica obtenida del análisis estocástico. Como se puede observar, la curva predicha en el análisis se ajusta perfectamente a las observaciones experimentales de la simulación.



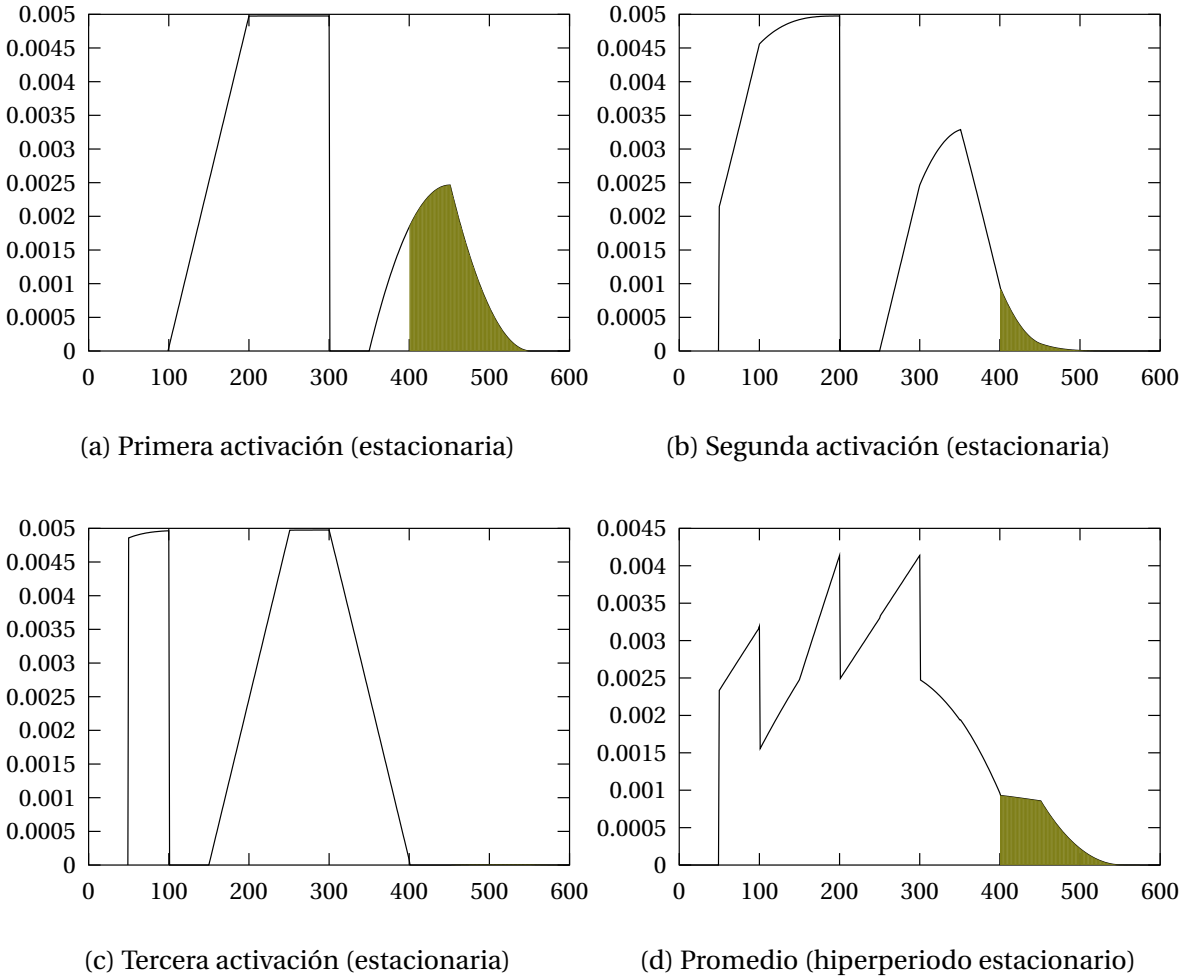
(a) Primera activación

(b) Segunda activación

(c) Tercera activación

(d) Promedio

Figura 6.21.: Funciones exactas de probabilidad del tiempo de respuesta de las tres primeras activaciones de τ_2 en el sistema S_1, y distribución promedio



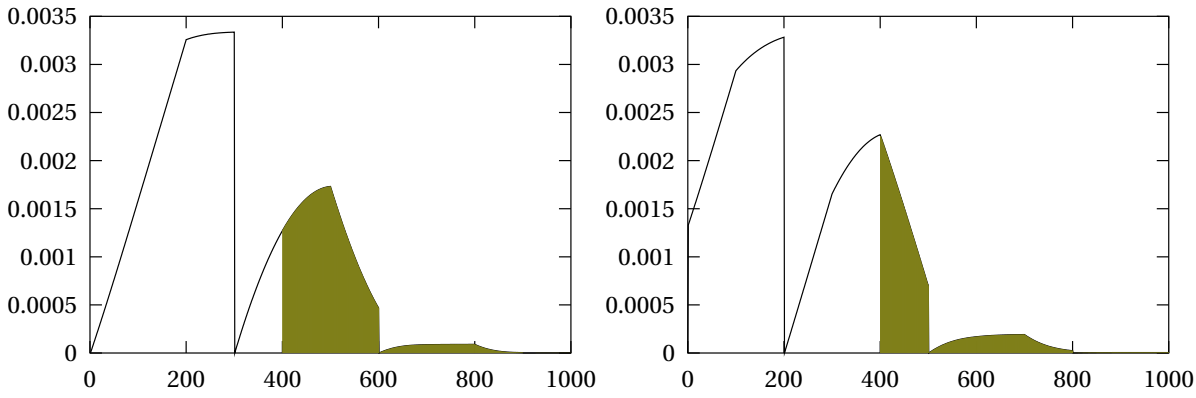
(a) Primera activación (estacionaria)

(b) Segunda activación (estacionaria)

(c) Tercera activación (estacionaria)

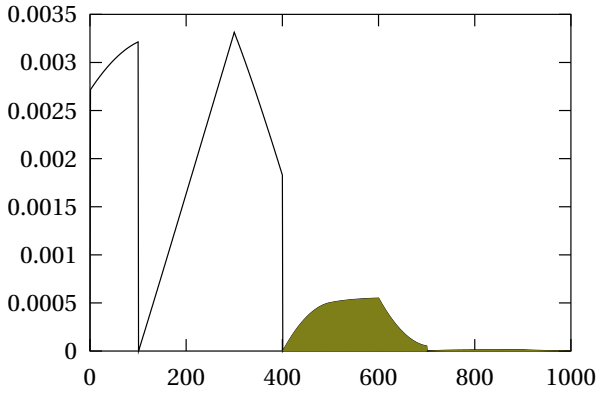
(d) Promedio (hiperperiodo estacionario)

Figura 6.22.: Funciones exactas de probabilidad del tiempo de respuesta de las tres primeras activaciones de τ_2 en el hiperperiodo estacionario del sistema S_2, y distribución promedio

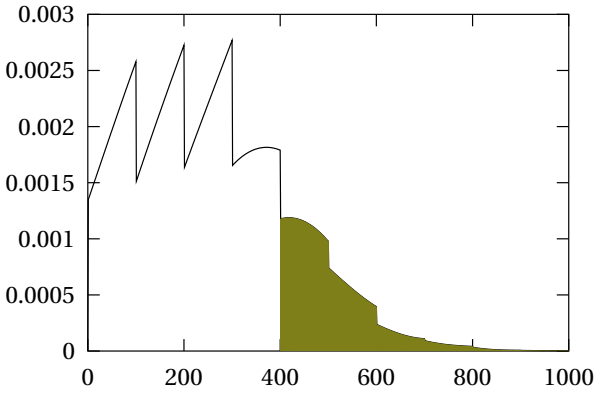


(a) Primera activación (estacionaria)

(b) Segunda activación (estacionaria)

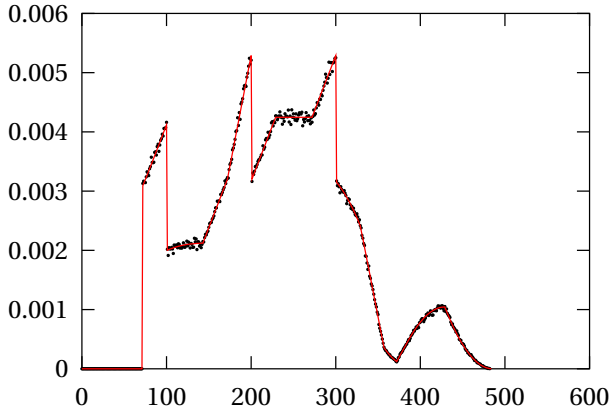


(c) Tercera activación (estacionaria)

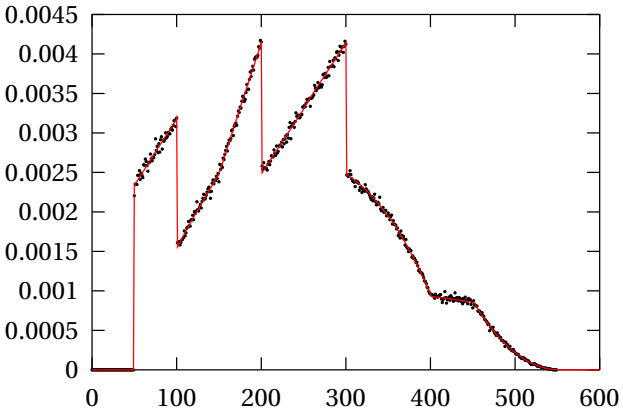


(d) Promedio (hiperperiodo estacionario)

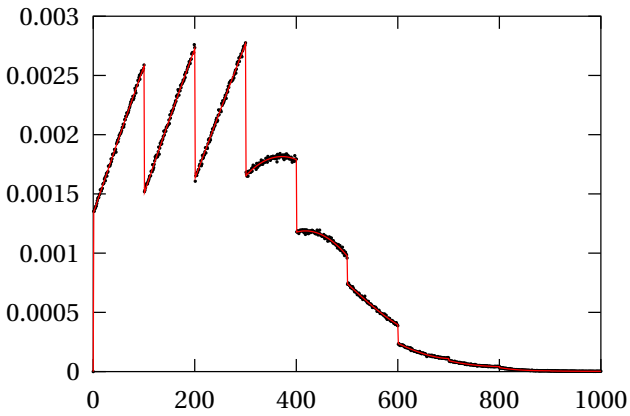
Figura 6.23.: Funciones exactas de probabilidad del tiempo de respuesta de las tres primeras activaciones de τ_2 en el hiperperiodo estacionario del sistema S_3, y distribución promedio



(a) Sistema S_1



(b) Sistema S_2



(c) Sistema S_3

Figura 6.24.: Histogramas de tiempos de ejecución de la tarea τ_2, comparados con la curva teórica obtenida del análisis

Simulación frente a análisis

Uno de los inconvenientes de la simulación es que la precisión de los resultados depende del tiempo durante el cuál se haya simulado. Para obtener las gráficas anteriores la simulación se mantuvo durante unos 330000 hiperperiodos para los sistemas S_1 y S_2. Para el sistema S_3, debido a que los tiempos presentan mucha mayor variabilidad, fue necesario prolongar la simulación durante unos dos millones y medio de hiperperiodos. En todo caso, con la potencia de un Pentium III, estas simulaciones requieren muy poco tiempo (5 segundos para cada sistema S_1 y S_2 y un minuto para el sistema S_3), por lo que cabe preguntarse si la simulación no podría ser un método alternativo de aproximación, evitando así todo el análisis estocástico.

Creo que esto no es una buena idea por dos razones:

- Incluso si los tiempos de simulación parecen cortos, lo cierto es que son varios órdenes de magnitud mayores que los que requiere el análisis estocástico. Los sistemas S_1 y S_2 son solucionados por análisis en apenas 5 milésimas de segundo, por lo que la simulación ha requerido en realidad 1000 veces más tiempo. Y aunque los resultados de la simulación ajustan bastante bien con la curva teórica, lo cierto es que no son demasiado fiables aún. Para un mejor ajuste la simulación debería ejecutarse durante mucho más tiempo.

- La simulación produce un buen ajuste en los tiempos de respuesta que aparecen a menudo, pero el ajuste es malo en los tiempos de respuesta con probabilidad baja. Esto es inherente al propio mecanismo de simulación. Los tiempos de respuesta con baja probabilidad aparecerán muy pocas veces (incluso ninguna) durante la simulación, por lo que la estimación de su frecuencia relativa tendrá siempre un error.

  Este segundo inconveniente es especialmente grave, ya que los tiempos de respuesta altos suelen tener baja probabilidad (obsérvese que en todas las gráficas la curva tiene una cola que tiende a cero), por lo que las probabilidades de estos tiempos no pueden obtenerse con precisión mediante simulación. Y son precisamente estos tiempos los que interesan al analista, ya que de ellos depende la probabilidad de pérdida de plazos.

Además, cuando la utilización $U^{\max}$ es mayor que uno, existe una probabilidad no nula de que parte de la carga generada en un hiperperiodo pase como carga inicial al hiperperiodo siguiente. La probabilidad de que esto ocurra es normalmente baja, y por tanto para que sea observable mediante simulación es necesario generar un gran número de hiperperiodos. Es decir, para sistemas con $U^{\max} > 1$, la simulación debe ejecutarse durante mucho más tiempo para tener resultados fiables.

Apoyaremos estas palabras con ejemplos. El sistema S_3 analizado en la sección anterior tiene $U^{\max} = 1{,}4108$. Su resolución mediante el análisis propuesto en esta tesis necesitó aproximadamente medio segundo para determinar la PF del tiempo de respuesta del trabajo $\Gamma_{2,1}$. A la simulación le hemos dado un tiempo 10 veces mayor (5 segundos). Si comparamos las PF obtenidas en cada caso (figura 6.25) veremos que evidentemente las curvas tienen la misma forma general, pero el ajuste no es demasiado bueno. Si mantenemos la simulación durante un minuto (del orden de 100 veces más del tiempo requerido por el análisis), el ajuste mejora (figura 6.26).

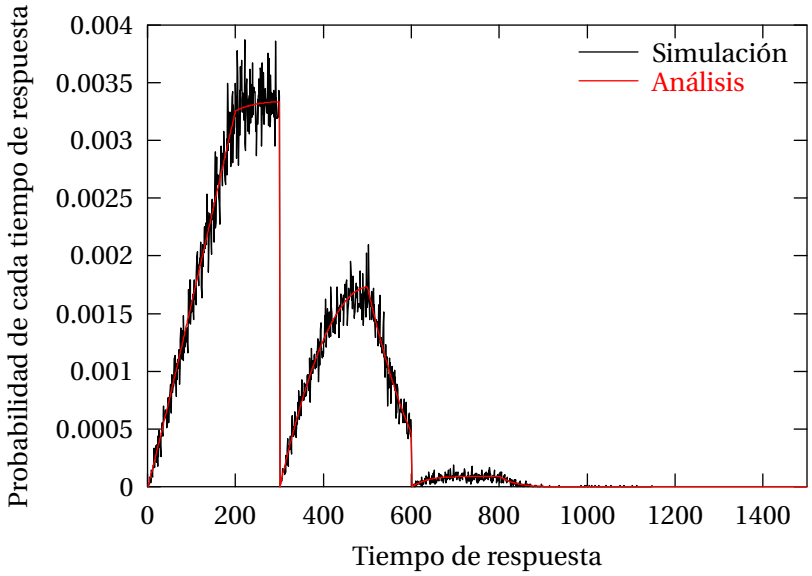


Figura 6.25.: Comparación entre el perfil del tiempo de respuesta obtenido por análisis y por simulación, cuando el tiempo de simulación es 10 veces el del análisis

Los puntos en los que hay mayor diferencia entre la simulación y el análisis serían los que tienen una probabilidad muy baja (por ejemplo, los tiempos de respuesta por encima de 1000). Lamentablemente la escala de la gráfica hace imposible discernir el valor de estos puntos. Por ello, vamos a representar la probabilidad de perder un plazo dado, en lugar de la probabilidad de un tiempo de respuesta. Para un plazo dado D, la probabilidad de perderlo sería el área que la gráfica anterior deja a la derecha de D, o lo que es lo mismo, 1 menos el área que queda a la izquierda. Representaremos este valor en escala logarítmica para que las diferencias sean más apreciables. Los resultados se muestran en la figura 6.27 (para un tiempo de simulación de 5 segundos) y 6.28 (para un tiempo de simulación de 1 minuto).

Aquí es claramente visible que la simulación es incapaz de estimar correctamente la probabilidad de perder un plazo, cuando el plazo es grande, debido co-

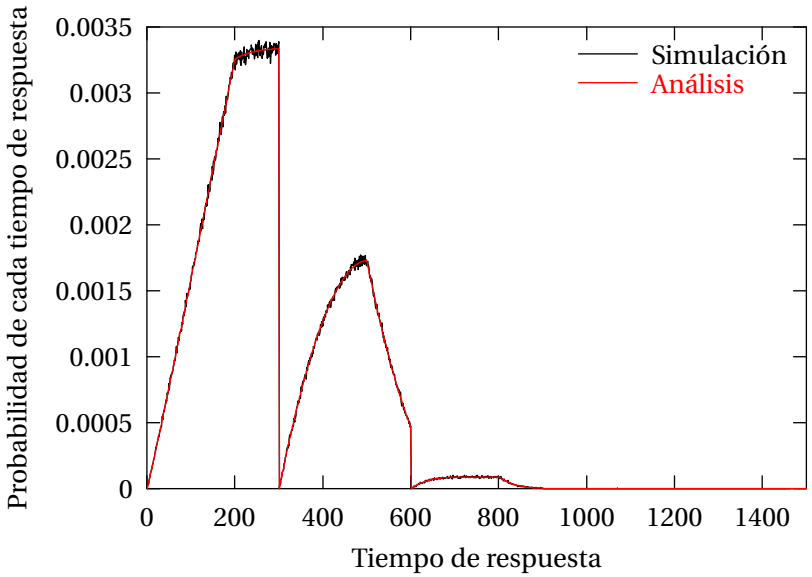


Figura 6.26.: Comparación entre el perfil del tiempo de respuesta obtenido por análisis y por simulación, cuando el tiempo de simulación es 100 veces el del análisis

mo hemos explicado a que estos casos son muy improbables y no aparecen a menos que la simulación se lleve a cabo durante largos lapsos de tiempo. Vemos que para una simulación corta (5 segundos) la estimación deja de ser fiable para plazos superiores a 800. Prolongando la simulación durante un minuto, mejora el ajuste a partir de 800, pero deja de ser fiable de nuevo a partir de 1100.

Estos resultados muestran que la simulación no es un buen método de aproximar las probabilidades de pérdida de plazo. Debe tenerse en cuenta además que en los ejemplos anteriores se ha supuesto un tiempo de ejecución con distribución uniforme. Si tenemos en cuenta que en la realidad el tiempo de ejecución tiene más bien forma de campana asimétrica, resulta que los valores extremos son muy improbables, lo que empeora aún más la estimación por simulación.

6.3.3. Conclusiones de las comparaciones

El método aproximado de Tia *et al.* [1995] proporciona una estimación pesimista de la probabilidad de perder el plazo. El pesimismo puede resultar excesivo, y no proporciona la función de probabilidad del tiempo de respuesta.

El método de Gardner [1999] proporciona otra aproximación de la probabilidad de perder plazos, obtenida a partir de la función de probabilidad del tiempo de respuesta. Al considerar únicamente la peor probabilidad de entre todas las activaciones, en lugar de promediarlas, la estimación es demasiado pesimista. Por otro lado, el algoritmo con el que calcula la PF del tiempo de respuesta de toda ac-

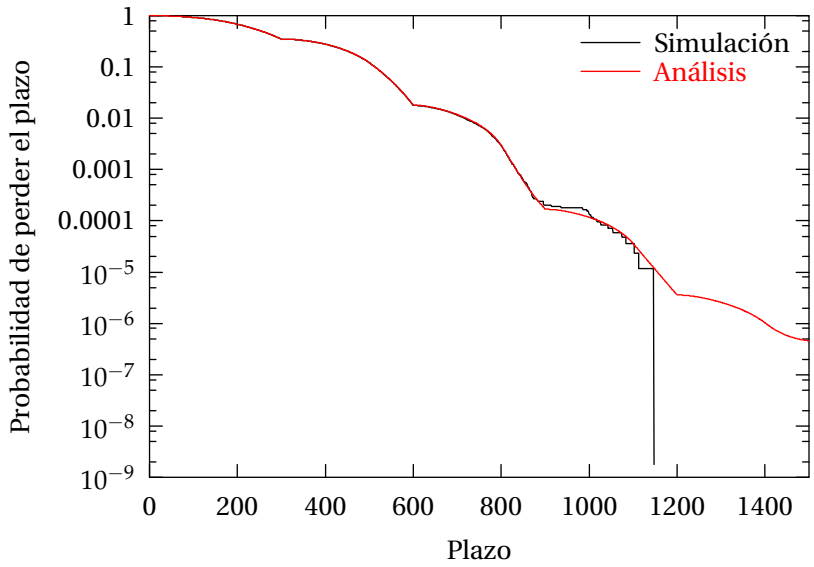


Figura 6.27.: Comparación entre la probabilidad de pérdida de plazo obtenida por análisis y por simulación, cuando el tiempo de simulación es 10 veces el del análisis

tivación distinta de la primera parece ser erróneo, puesto que los resultados que Gardner obtiene y publica son aún más pesimistas de lo que deberían ser.

El análisis estocástico propuesto en esta tesis proporciona la probabilidad exacta y no una aproximación. Esto es corroborado por los resultados obtenidos mediante simulación. De hecho, el análisis es superior a la simulación por requerir menos tiempo de cómputo y dar resultados mucho más precisos incluso cuando las probabilidades de perder plazo son muy bajas. Este tipo de probabilidades tan bajas no se pueden obtener mediante simulación, y son precisamente las más interesantes en el diseño de sistemas de tiempo real con plazos firmes.

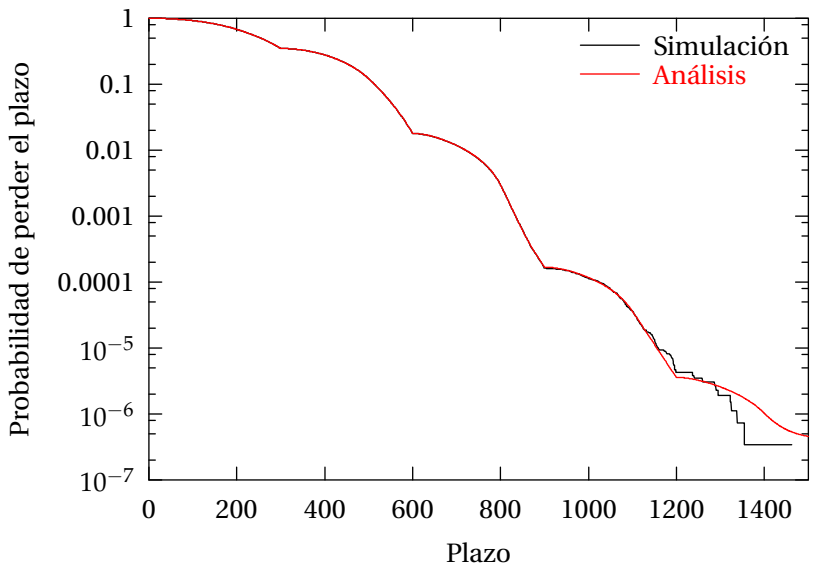


Figura 6.28.: Comparación entre la probabilidad de pérdida de plazo obtenida por análisis y por simulación, cuando el tiempo de simulación es 100 veces el del análisis

7. Conclusiones

7.1. Objetivos alcanzados y aportaciones

El análisis propuesto en esta tesis permite modelar un sistema de tareas periódicas en el que los tiempos de ejecución son variables aleatorias que vienen especificadas por su función de probabilidad, y derivar de este modelo las funciones de probabilidad de los tiempos de respuesta de cada una de las tareas, de los cuales a su vez se puede obtener las probabilidades de perder los plazos. El análisis es aplicable a planificadores basados en prioridades, como los clásicos RM, DM o EDF, y admite tanto tareas con expulsión como sin expulsión. El modelo puede ampliarse para tomar en consideración el bloqueo por acceso a recursos compartidos y el *jitter*, y puede combinarse con servidores de tareas aperiódicas con reserva de ancho de banda.

Las aportaciones más importantes de esta tesis, con respecto a otros trabajos que han abordado un problema similar, son las siguientes:

- El modelo no impone restricciones sobre los plazos de las tareas, que pueden ser menores, iguales o mayores que los periodos, sin que esto tenga influencia sobre el análisis.

- La utilización máxima del sistema puede ser mayor que 1, y aún así es posible encontrar la distribución del tiempo de respuesta de las tareas para el "régimen permanente" del sistema.

- No se requiere un planificador especial, el algoritmo de planificación asumido es uno basado en prioridades, sin que importe la política de asignación de las mismas, con tal de que las prioridades se asignen de forma determinista y predecible fuera de línea (ej: RM, DM, EDF).

- Se proporcionan demostraciones matemáticas rigurosas de las condiciones bajo las cuales las probabilidades "convergen" hacia una distribución estacionaria. Esto no había sido realizado en ninguno de los trabajos previos, que daban por hecho que esta convergencia ocurriría cuando la utilización promedio fuera menor que 1, pero sin ofrecer demostración de esta afirmación.

- Se identifica el caso $U^{\max} < 1$ como especial, en el sentido de que se puede resolver con un análisis mucho más sencillo que el caso general, y se ofrece el método para este caso.

- Para el caso general con $U^{\max} \geq 1$, el análisis se basa en un modelado mediante cadenas de Markov, muy parecido al de Kim *et al.* [2002], si bien ha sido desarrollado de forma independiente. De todas formas, en esta tesis se ofrecen aportaciones adicionales sobre el citado trabajo:

  - El método para obtener la matriz de Markov que modela la evolución del sistema es mucho más simple y eficiente.
  - Se ofrece un método para obtener de forma exacta la distribución estacionaria, que no requiere la truncación de la matriz de Markov. Este método permite conocer la expresión de la forma general que tiene la solución estacionaria para cualquier sistema, y puede usarse para "resolver" sistemas no muy grandes.
  - El caso con expulsión no requiere un tratamiento especial, ni la agrupación de trabajos. Una vez obtenida la distribución estacionaria de la carga, de ella se derivan los tiempos de respuesta tanto para tareas con expulsión como sin ella.
  - El método es aplicable también a RM, además de EDF, y de hecho a cualquier mecanismo de asignación de prioridades que presente una periodicidad en cuanto a las prioridades relativas de las tareas.

- Se introduce la idea de "distribución peor" que permite comparar dos variables aleatorias y decidir cuál de ellas es "peor" en el contexto del tiempo real. Esta idea es rigurosamente definida y sirve de base para hacer "aproximaciones conservadoras", es decir, métodos de análisis aproximados que garanticen que la probabilidad real de perder el plazo será siempre menor que la que se obtiene del análisis.

- El modelo básico se extiende para tratar el acceso a recursos compartidos y el *jitter*, si bien en este caso las probabilidades obtenidas ya no son exactas, sino "peores", de acuerdo con la definición de "distribución peor".

- Se realiza un detallado estudio de la complejidad computacional del análisis, y se ofrecen medidas experimentales del tiempo requerido para resolver diferentes tipos de sistemas, en función de sus parámetros. Con esto se ofrece una idea del rango práctico de aplicabilidad del análisis propuesto.

7.2. Publicaciones derivadas de la tesis

El modelo básico de sistema y la solución basada en los algoritmos "convolucionar y encoger" y "partir, convolucionar y juntar" fueron presentados por primera vez en público en febrero de 2002, en las *V Jornadas de Tiempo Real*, una reunión de investigadores en el campo del tiempo real, de ámbito nacional[1].

Las mismas ideas fueron publicadas en un foro internacional en Octubre de 2002, en el *1st CARTS Workshop on Advanced Real-Time Technologies*[2].

Estas ideas, ampliadas con el tratamiento del caso $U^{\max} > 1$, es decir, el análisis Markoviano del hiperperiodo estacionario, fueron enviadas al *23rd IEEE International Real-Time Systems Symposium*, donde coincidieron con otro trabajo de similares características desarrollado independiemente por Kim, Lo Bello *et al.* Dada la similitud del problema abordado, y de algunos aspectos de la solución, el comité técnico nos sugirió la posibilidad de colaborar y realizar un artículo conjunto, fusionando ambos enfoques. Así lo hicimos y el resultado fue publicado en Diciembre de 2002[3].

La colaboración así iniciada continúa en la actualidad, y actualmente (Mayo de 2003) estamos en el proceso de finalizar un artículo, con la intención de publicarlo en revista, que resumirá las ideas publicadas en [Díaz *et al.*, 2002*a*], ampliando la base teórica del método "retroceder y repetir" (aportación de Kim), y añadiendo el estudio de la complejidad computacional junto con los resultados experimentales asociados a ella (aportación mía).

7.3. Limitaciones y ampliaciones

La principal limitación del análisis se halla en su coste computacional, que es muy elevado. Aunque se ha mostrado que para sistemas relativamente grandes (del orden del millar de activaciones de tareas por hiperperiodo) los tiempos de análisis pueden ser razonables (del orden de segundos), no obstante el crecimiento del coste es cúbico, haciendo que resulte prohibitivo para sistemas muy grandes.

Otra limitación del modelo está en la hipótesis de independencia estadística entre las tareas. Algunos autores sostienen que esta hipótesis no es válida en muchos casos, ya que se observa una cierta correlación entre los tiempos de diferentes tareas, de modo que si una de ellas tarda mucho, es más probable que la siguiente también tarde mucho. En principio el modelo se puede ampliar para incluir esta información (en lugar de tener una función de probabilidad para cada tarea, podríamos tener por ejemplo una distribución conjunta para cada par de tareas).

[1] Referencia [Díaz *et al.*, 2002*b*]
[2] Referencia [Díaz *et al.*, 2002]
[3] Referencia [Díaz *et al.*, 2002*a*]

El problema es que los requisitos computacionales y de memoria crecerían aún más.

Tenemos pues dos líneas de investigación diferentes, aunque relacionadas, por las cuales se podría continuar ampliando el método:

- Búsqueda de algoritmos más eficientes o simplificaciones que disminuyan la complejidad computacional. En lo posible debería buscarse que estas simplificaciones sean "conservadoras", haciendo uso de la propiedad "peor que".

  Por ejemplo, el método iterativo puede ser perfeccionado para garantizar que en cualquiera de sus iteraciones la solución obtenida es "peor que" la exacta.

  También se puede investigar en el ratio de convergencia de un sistema hacia su distribución estacionaria, para poder estimar de antemano si la aproximación iterativa merece la pena frente a otras técnicas.

- Ampliación del modelo para eliminar algunas de sus limitaciones y tomar en cuenta información adicional que el modelo actual no captura. El problema es que aumentar el modelo también aumentará su complejidad, por lo que es necesario que estas nuevas ampliaciones vayan acompañadas de simplificaciones para que el problema siga siendo tratable. Estas simplificaciones también deberían ser "conservadoras".

Anexos

A. Convolución discreta

A.1. Definición

La convolución de dos funciones discretas $f(x)$ y $g(x)$ se denota por $(f \otimes g)(x)$ y se define así:

$$(f \otimes g)(x) = \sum_{i=-\infty}^{\infty} f(i)g(x-i) \tag{A.1}$$

A.2. Implementación

En nuestro caso en que las funciones representan probabilidades de tiempos de ejecución, su valor mínimo será cero, y su valor máximo estará acotado, por lo que la función $f(x)$ se puede implementar como un vector $f[x]$. La variable x actúa como índice del vector y el valor almacenado en $f[x]$ sería la probabilidad de x. En este caso la ecuación A.1 se reescribe así:

$$(f \otimes g)[x] = \sum_{i=0}^{x^{\max}} f[i] \cdot g[x-i] \tag{A.2}$$

Un algoritmo directo para implementar la convolución es el siguiente (se asume que f y g contienen elementos indexados desde cero):

```
1: CONVOLUCIÓN(f, g)
2: Reservar memoria para (Longitud(f)+Longitud(g)−1) elementos en un array
   que llamaremos r.
3: Inicializar todos los elementos de r con cero
4: para i = 0,...,Longitud(f)−1 hacer
5:     para j = 0,...,Longitud(g)−1 hacer
6:         r[i + j] ← r[i + j] + f[i] · g[j]
7:     fin para
8: fin para
9: Retornar r
```

Se deduce del algoritmo que para computar esta convolución, si el vector f tiene m_1 elementos y el vector g tiene m_2 elementos, son necesarias $(m_1 m_2)$ multiplicaciones y $(m_1 m_2)$ sumas, y por tanto $2m_1 m_2$ operaciones en total. El vector resultante de la convolución tendrá $(m_1 + m_2 - 1)$ elementos. Si los dos vectores tienen un tamaño aproximadamente igual, n, la complejidad del algoritmo es $O(n^2)$.

A.2.1. Optimización

La convolución también puede calcularse a través de la transformada de Fourier. Si f' es la transformada de Fourier del vector f, g' la de g y $(f \otimes g)'$ la de $(f \otimes g)$, se cumple:

$$(f \otimes g)'[x] = f'[x]g'[x] \tag{A.3}$$

En este caso, dadas las transformadas, se requieren simplemente $\max(m_1, m_2)$ multiplicaciones. Sin embargo calcular las transformadas de Fourier es un proceso costoso, por lo que la complejidad al final sería la misma que por el método directo.

Si los tamaños de los vectores f y g son potencias de dos, en ese caso se puede utilizar la transformada rápida de Fourier (FFT) para optimizar la conversión. Si no son potencia de dos, en cualquier caso se pueden rellenar con ceros por la derecha para que lo sean. Si ambos vectores tienen aproximadamente el mismo tamaño, n, el coste total de aplicar la transformada, realizar las n multiplicaciones, y aplicar la anti-transformada para obtener el resultado de la convolución es $O(n \log n)$, y por tanto parece ventajoso frente al $O(n^2)$ de la implementación directa.

Sin embargo, si los tamaños de los vectores f y g no son similares, para poder aplicar la FFT necesitamos rellenar con ceros ambos hasta que tengan el mismo tamaño (y además éste sea potencia de 2). En este caso la complejidad de realizar la convolución usando la FFT sería $O(N \log N)$ siendo $N = 2^{\lceil \log_2 \max(n,m) \rceil}$. Esto puede ser mayor que el coste del método directo, que es simplemente $O(mn)$.

En nuestro problema es muy frecuente que las funciones a convolucionar tengan tamaños dispares, ya que una de ellas suele ser el PF de la carga (que tiene un gran número de puntos) y el otro el PF del tiempo de ejecución de una tarea, que tiene muchos menos. Por esta razón para nuestro problema particular no interesa una implementación basada en FFT.

A.2.2. Funciones de probabilidad acumuladas

Como es sabido, para obtener la función de probabilidad (PF) de la suma de dos variables aleatorias, basta hallar la convolución discreta de las PFs de éstas

variables. Sin embargo, la función de probabilidad acumulada (CDF) de la misma suma, no es igual a la convolución de las CDFs, sino a la convolución de la CDF de una con la PF de la otra. Esto se demuestra en el siguiente teorema.

Teorema 13. *Si $F_{\mathcal{X}}$ es la función de probabilidad acumulada (CDF) de una variable aleatoria $\mathcal{X}$ y $f_{\mathcal{X}}$ su función de probabilidad (PF), y análogamente $F_{\mathcal{Y}}$ y $f_{\mathcal{Y}}$ para la variable aleatoria $\mathcal{Y}$, la CDF de la suma de ambas variables se calcula como:*

$$F_{\mathcal{X}+\mathcal{Y}}(x) = (F_{\mathcal{X}} \otimes f_{\mathcal{Y}})(x) = (f_{\mathcal{X}} \otimes F_{\mathcal{Y}})(x) \tag{A.4}$$

Demostración. El valor de $F_{\mathcal{X}+\mathcal{Y}}(x)$ puede obtenerse a partir de la función $f_{\mathcal{X}+\mathcal{Y}}$, por definición, en la forma siguiente:

$$F_{\mathcal{X}+\mathcal{Y}}(x) = \sum_{j=-\infty}^{x} f_{\mathcal{X}+\mathcal{Y}}(j)$$

Por otro lado $f_{\mathcal{X}+\mathcal{Y}}(\cdot)$ puede obtenerse convolucionando $f_{\mathcal{X}}$ con $f_{\mathcal{Y}}$, y aplicando la fórmula de la convolución (A.1):

$$F_{\mathcal{X}+\mathcal{Y}}(x) = \sum_{j=-\infty}^{x} (f_{\mathcal{X}} \otimes f_{\mathcal{Y}})(j) = \sum_{j=-\infty}^{x} \sum_{i=-\infty}^{\infty} f_{\mathcal{X}}(i) \cdot f_{\mathcal{Y}}(j-i)$$

Si cambiamos de orden los sumatorios

$$F_{\mathcal{X}+\mathcal{Y}}(x) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{x} f_{\mathcal{X}}(i) \cdot f_{\mathcal{Y}}(j-i)$$

y ya que $f_{\mathcal{X}}(i)$ no depende del índice j del sumatorio interno, se le puede sacar del mismo:

$$F_{\mathcal{X}+\mathcal{Y}}(x) = \sum_{i=-\infty}^{\infty} f_{\mathcal{X}}(i) \sum_{j=-\infty}^{x} f_{\mathcal{Y}}(j-i)$$

El sumatorio interno nos da la probabilidad de que $\mathcal{Y}$ sea menor o igual que $x-i$, lo que por definición es $F_{\mathcal{Y}}(x-i)$, por tanto:

$$F_{\mathcal{X}+\mathcal{Y}}(x) = \sum_{i=-\infty}^{\infty} f_{\mathcal{X}}(i) \cdot F_{\mathcal{Y}}(x-i)$$

Con lo que finalmente llegamos en el segundo término a una expresión que reconocemos como $(f_{\mathcal{X}} \otimes F_{\mathcal{Y}})(x)$. De forma análoga, se demuestra que también es igual a $(f_{\mathcal{Y}} \otimes F_{\mathcal{X}})(x)$, puesto que la suma de variables aleatorias es conmutativa. □

B. Demostraciones adicionales

B.1. Número de raíces con módulo mayor de 1 en el polinomio característico de la matriz A

En la sección 4.5.3 se muestra como se puede hallar la distribución estacionaria completa de forma exacta, resolviendo la ecuación $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$. El método consiste en plantear un conjunto de $m_r + 1$ ecuaciones que provienen de las primeras $m_r + 1$ filas de $\mathbf{P}$, más otras $r - 1$ ecuaciones que deben provenir de una relación de recurrencia, siendo r el índice de la primera columna en $\mathbf{P}$ donde comienza su regularidad, y m_r el índice del último elemento no nulo en dicha columna.

Usando aquel método, el número total de incógnitas siempre será $m_r + r + 1$. Las $m_r + 1$ primeras filas de $\mathbf{P}$ siempre proporcionan $m_r + 1$ ecuaciones linealmente independientes. Por otro lado, tenemos la condición adicional de que la suma de todas las componentes de $\boldsymbol{\pi}$ debe ser 1. Así que sólo faltan $r - 1$ ecuaciones para completar el sistema. Si pudiéramos garantizar que la relación de recurrencia siempre proporcionará $r - 1$ ecuaciones adicionales, tendríamos el sistema completo.

Demostraremos en este anexo que, efectivamente, se pueden conseguir $r - 1$ ecuaciones adicionales de la relación de recurrencia, siempre que el sistema cumpla la condición de estabilidad $\bar{U} < 1$. De hecho, se consiguen r ecuaciones, pero una de ellas siempre es redundante, puesto que corresponde al auto-valor 1 que siempre está presente.

Recordemos de dónde provienen estas r ecuaciones extra. La relación de recurrencia podía expresarse en forma matricial, véase la eq. (4.29), donde la matriz $\mathbf{A}$ tenía una estructura muy regular, con ceros en todos sus elementos, salvo una diagonal superior con unos, y salvo la última fila, véase la eq. (4.28), en la pág. 109. Para resolver la relación de recurrencia matricial era necesario diagonalizar la matriz $\mathbf{A}$, lo que implicaba calcular sus valores propios. Al hacerlo, se descubría que alguno de estos valores propios era mayor o igual a 1 y estos debían desaparecer de la solución final, pues de lo contrario los componentes de $\boldsymbol{\pi}$ no serían sumables (su suma sería infinita). En concreto, aparecían r valores propios con módulo mayor o igual que 1, y al forzar que los correspondientes coeficientes fuesen cero, se obtenían las r ecuaciones adicionales.

De modo que se trata de garantizar que el polinomio característico de $\mathbf{A}$ siempre tiene r raíces con módulo mayor o igual que uno, si se cumple la condición de

estabilidad $\bar{U} < 1$.

Ya que la última línea de **A** es la que nos da el polinomio característico, y esta última línea se construye a partir de los coeficientes $b_r(\cdot)$ de la columna r-ésima de **P**, podemos ver que en realidad el polinomio característico depende sólo de estos coeficientes. Es sencillo demostrar que el polinomio característico de **A** responde a la forma:

$$f(\lambda) = \big(\sum_{i=0}^{m_r} b_r(i)\lambda^{m_r-i}\big) - \lambda^{m_r-r} = 0 \tag{B.1}$$

Por otro lado, se demostró en la proposición 3 de la página 97 que la condición $\bar{U} < 1$ equivale a $\sum_i i b_r(i) < r$, lo cual es una restricción sobre los coeficientes del polinomio característico. El siguiente teorema demostrará que si se cumple esta restricción, el polinomio tiene r raíces de modulo mayor o igual que 1.

Teorema 14. *Un polinomio $f(z)$, de grado m_r, de la forma*

$$f(z) = \sum_{i=0}^{m_r} a_i z^{m_r-i} - z^{m_r-r} = 0$$

en el que los coeficientes a_i cumplen las siguientes restricciones:

$$0 \le a_i < 1; a_{m_r} \neq 0; \qquad \sum_{i=0}^{m_r} a_i = 1; \qquad \sum_{i=0}^{m_r} i a_i < r$$

tiene exactamente r raíces de módulo mayor o igual a la unidad.

Demostración. La demostración se basará en el teorema de Rouché, que es un resultado clásico del análisis de variable compleja [Rudin, 1966]. Este teorema dice que, si existe otra función $g(z)$ y un contorno cerrado C tales que:

$$|f(z) + g(z)| < |g(z)| \quad \forall z \in C \tag{B.2}$$

entonces $f(z)$ y $g(z)$ tienen el mismo número de raíces en la región delimitada por C.

Para demostrar el teorema que nos ocupa, se tomará $g(z) = z^{m_r-r}$, y C igual al círculo centrado en el origen de radio $1-\epsilon$, con $\epsilon > 0$ muy pequeño.

Utilizando esta $g(z)$ se tiene que $f(z) + g(z) = \sum_{i=0}^{m_r} a_i z^{m-i}$. En cuanto a su módulo, se puede aplicar la desigualdad triangular por ser todos los a_i positivos:

$$|f(z) + g(z)| \le \sum_{i=0}^{m_r} a_i |z|^{m_r-i}$$

En particular, para los $z \in C$, también se cumplirá, luego:

$$|f(z)+g(z)| \leq \sum_{i=0}^{m_r} a_i |1-\epsilon|^{m_r-i} \qquad z \in C$$

El segundo miembro es igual a la función $f(z)+g(z)$ evaluada en el real $(1-\epsilon)$, por lo que:

$$|f(z)+g(z)| \leq f(1-\epsilon)+g(1-\epsilon) = (f+g)(1-\epsilon) \qquad z \in C \tag{B.3}$$

Pero además $f(1-\epsilon)+g(1-\epsilon)$ es menor que $g(1-\epsilon)$. Esto se debe a dos hechos:

1. Las funciones $f+g$ y g valen ambas 1 para $z=1$, es decir $(f+g)(1) = g(1) = 1$, como puede fácilmente comprobarse.
2. La derivada de $f+g$ en 1 es mayor que la de g. En efecto:

$$(f+g)'(1) = \sum_{i=0}^{m_r}(m_r-i)a_i = m_r - \sum_{i=0}^{m_r} i a_i > m_r - r$$
$$g'(1) = m_r - r$$

Al ser la pendiente de $f+g(z)$ mayor que la de $g(z)$ en $z=1$, $f+g$ ha de ser menor que g en $(1-\epsilon)$, como se muestra en la figura B.1.

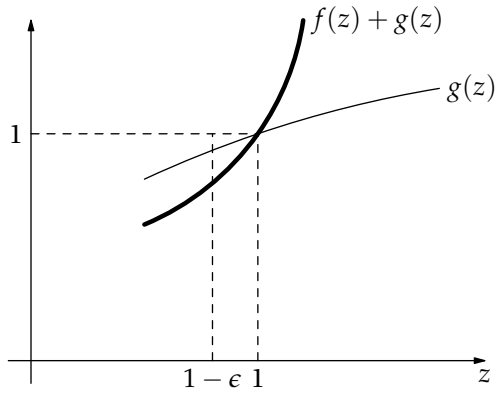


Figura B.1.: Ilustración de que $f+g < g$ a la izquierda de 1

Por tanto $(f+g)(1-\epsilon) < g(1-\epsilon)$, y llevando esto a la eq. (B.3), y teniendo en cuenta que $g(1-\epsilon)$ es positiva por ser próxima a 1, tenemos finalmente:

$$|f(z)+g(z)| < g(1-\epsilon) = |g(1-\epsilon)| \qquad z \in C$$

Es decir, se cumplen las condiciones del teorema de Rouché, luego f tiene tantas raíces como g dentro de C. Y ya que las $m_r - r$ raíces de g están dentro de C (pues son todas cero), se sigue que f tiene $m_r - r$ raíces de módulo menor a 1, y por tanto r raíces de módulo mayor o igual que 1. □

B.2. Propiedades de la relación "peor que"

En la página 132, definición 13 se definió una relación de ordenación entre variables aleatorias. Repetimos la definición para comodidad del lector.

Dadas dos variables aleatorias $\mathcal{A}$ y $\mathcal{B}$, diremos que $\mathcal{A}$ es **peor** que $\mathcal{B}$, y lo denotaremos por $\mathcal{A} \succcurlyeq \mathcal{B}$, si entre sus CDFs se cumple la siguiente relación:

$$F_{\mathcal{A}}(i) \leq F_{\mathcal{B}}(i) \quad \forall i$$

Esta relación tiene una serie de propiedades que se enunciaron sin demostración en el capítulo 5, y que se demuestran en este apéndice.

Proposición 8. *$\mathcal{A} \succcurlyeq \mathcal{A}$ (propiedad reflexiva)*

Demostración. Trivialmente, ya que $F_{\mathcal{A}}(\cdot) \leq F_{\mathcal{A}}(\cdot)$ en todos sus puntos. □

Proposición 9. *Si $\mathcal{A} \succcurlyeq \mathcal{B}$ y $\mathcal{B} \succcurlyeq \mathcal{C}$, entonces $\mathcal{A} \succcurlyeq \mathcal{C}$ (propiedad transitiva)*

Demostración. Por hipótesis, para cualquier i se cumple $F_{\mathcal{A}}(i) \leq F_{\mathcal{B}}(i)$, y también $F_{\mathcal{B}}(i) \leq F_{\mathcal{C}}(i)$, luego es cierto que $F_{\mathcal{A}}(i) \leq F_{\mathcal{C}}(i)$. □

Proposición 10. *Si $\mathcal{A} \succcurlyeq \mathcal{B}$ y $\mathcal{C}$ es positiva, esto es $\mathcal{C} \succcurlyeq \mathbb{0}$, entonces $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{C}$ (conservación de la relación "peor que" con la suma).*

Demostración. La CDF de $\mathcal{A} + \mathcal{C}$ puede calcularse mediante convolución de la CDF de $\mathcal{A}$ con la PF de $\mathcal{C}$ (teorema 13), y lo mismo para la CDF de $\mathcal{B} + \mathcal{C}$.

$$F_{\mathcal{A}+\mathcal{C}}(x) = (F_{\mathcal{A}} \otimes f_{\mathcal{C}})(x) = \sum_{i=-\infty}^{\infty} F_{\mathcal{A}}(i) \cdot f_{\mathcal{C}}(x-i)$$

$$F_{\mathcal{B}+\mathcal{C}}(x) = (F_{\mathcal{B}} \otimes f_{\mathcal{C}})(x) = \sum_{i=-\infty}^{\infty} F_{\mathcal{B}}(i) \cdot f_{\mathcal{C}}(x-i)$$

Ahora bien, $F_{\mathcal{A}}(i) \leq F_{\mathcal{B}}(i)$ por hipótesis, luego el primer sumatorio es menor que el segundo y $F_{\mathcal{A}+\mathcal{C}}(x) \leq F_{\mathcal{B}+\mathcal{C}}(x)$, por lo que $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{C}$. □

Proposición 11. *Si $\mathcal{A} \succcurlyeq \mathcal{B} \succcurlyeq \mathbb{0}$ y $\mathcal{C} \succcurlyeq \mathcal{D} \succcurlyeq \mathbb{0}$, entonces $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{D}$.*

Demostración. Al ser $\mathcal{C} \succcurlyeq \mathbb{0}$ y $\mathcal{A} \succcurlyeq \mathcal{B}$, se cumplirá que $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{C}$, por la propiedad antes demostrada. Por otro lado, al ser $\mathcal{B} \succcurlyeq \mathbb{0}$ y $\mathcal{C} \succcurlyeq \mathcal{D}$, se cumplirá también que $\mathcal{B} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{D}$ por la misma propiedad. Y juntando la primera conclusión con la segunda, por medio de la propiedad transitiva, se obtiene finalmente que $\mathcal{A} + \mathcal{C} \succcurlyeq \mathcal{B} + \mathcal{D}$. □

Proposición 12. *Si $\mathcal{B} \succcurlyeq \mathbb{0}$, entonces $\mathcal{A} + \mathcal{B} \succcurlyeq \mathcal{A}$ (incremento del pesimismo al sumar)*

Demostración. Ya que, por la propiedad reflexiva $\mathcal{A} \succcurlyeq \mathcal{A}$, y por hipótesis $\mathcal{B} \succcurlyeq \mathbb{O}$, sumando ambas inecuaciones tenemos $\mathcal{A} + \mathcal{B} \succcurlyeq \mathcal{A} + \mathbb{O}$ (esta suma mantiene la desigualdad en virtud de la propiedad antes demostrada). Por otro lado $\mathcal{A} + \mathbb{O} = \mathcal{A}$, como es fácil comprobar haciendo la convolución $(f_{\mathcal{A}} \otimes f_{\mathbb{O}})$ y teniendo en cuenta que por definición $f_{\mathbb{O}}$ es nula en todos sus puntos salvo en cero que vale 1. De todo esto se desprende que la proposición es cierta. □

C. Distribución beta

C.1. Definición

La distribución beta se suele usar para representar variables físicas, cuyos valores se encuentran restringidos a un intervalo de longitud finita y permite representar una gran variedad de perfiles [Canavos, 1988].

Se trata de una distribución para variables aleatorias continuas que toman valores únicamente en el intervalo $[0,1]$. La forma de la distribución depende de los valores que tomen sus dos parámetros, típicamente denominados α y β.

Su función de densidad viene dada por la fórmula:

$$f_{\text{beta}}(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{C.1}$$

donde $B(a,b)$ es la función beta, definida por:

$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{C.2}$$

siendo $\Gamma(x)$ la función Gamma.

C.2. Forma de la función de densidad

Cuando ambos parámetros son iguales, la distribución es simétrica. Si además $\alpha = \beta = 1$, se trata de una distribución uniforme. Para valores de $\alpha = \beta < 1$ la función tiene forma de "U", y para valores de $\alpha = \beta > 1$ tiene forma de "campana". La figura C.1 muestra la gráfica de su función de densidad para diferentes casos en los que $\alpha = \beta$.

Si $\alpha > \beta$, la curva es asimétrica, estando la masa desplazada hacia la derecha, y si $\alpha < \beta$, la masa se desplaza hacia la izquierda. Si ambos parámetros son menores de 1, la forma general es de una "U" asimétrica, con un brazo más largo que otro, por así decir. Si $\alpha < 1$ pero $\beta > 1$, la curva tiene una forma similar a la de una exponencial, en el sentido de que toma valores altos al principio y va decayendo hasta hacerse cero al final. Sin embargo, una exponencial tiene una cola infinita,

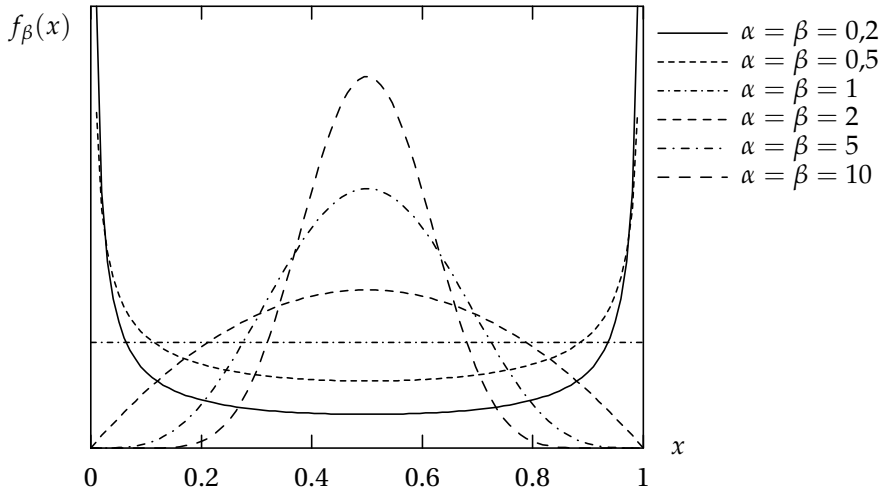


Figura C.1.: Diferentes formas de la distribución beta, cuando sus dos parámetros son iguales

mientras que la beta está limitada al intervalo $[0,1]$. Si ambos parámetros son mayores de 1, la curva tiene forma de campana asimétrica. Conforme aumentan los valores de α y β, la campana se hace más "afilada". Si $\alpha = 1, \beta = 2$ la distribución es triangular izquierda, y si $\alpha = 2, \beta = 1$ es triangular derecha.

Las figuras siguientes muestran diferentes formas que puede adoptar la función beta, según los valores de los parámetros α y β.

Como vemos, se trata de una distribución que resulta muy adecuada para modelar tiempos de ejecución, debido a dos razones:

- El rango de la variable aleatoria está acotado, al igual que los tiempos de ejecución están acotados. Aunque la distribución beta está limitada al intervalo $[0,1]$, éste puede ser fácilmente reescalado y desplazado para que cubra el rango $[C^{\min}, C^{\max}]$.
- Es una distribución muy versátil que permite modelar diferentes formas de los tiempos de ejecución. Entre ellas la uniforme y la campana desplazada a la izquierda. Esta última resulta especialmente adecuada, ya que como se sabe los tiempos de ejecución más probables suelen estar bastante alejados de los peores tiempos de ejecución, que tienen una probabilidad muy baja.

C.3. Propiedades

Algunas propiedades conocidas de la distribución beta son las siguientes.

- La media toma el valor $\dfrac{\alpha}{\alpha + \beta}$

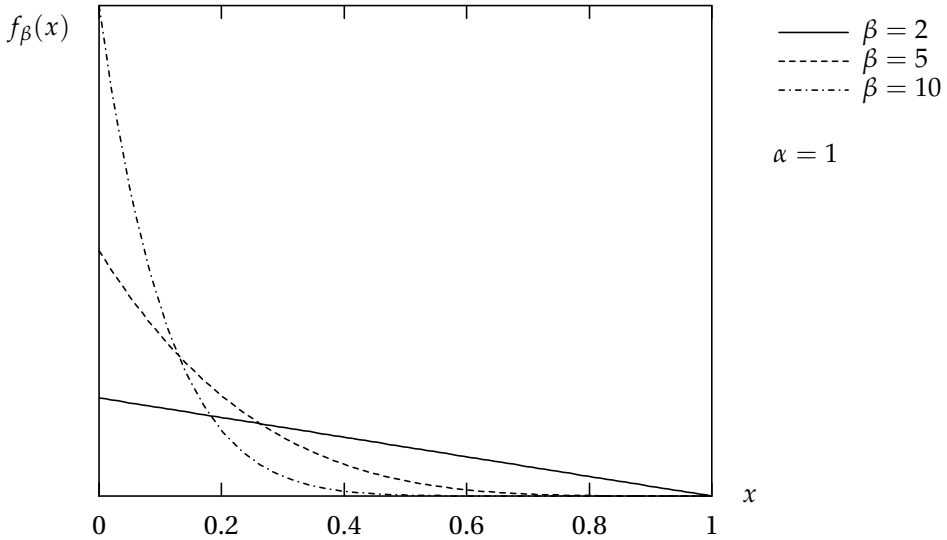


Figura C.2.: Diferentes formas de la distribución beta, cuando uno de sus parámetros toma el valor 1

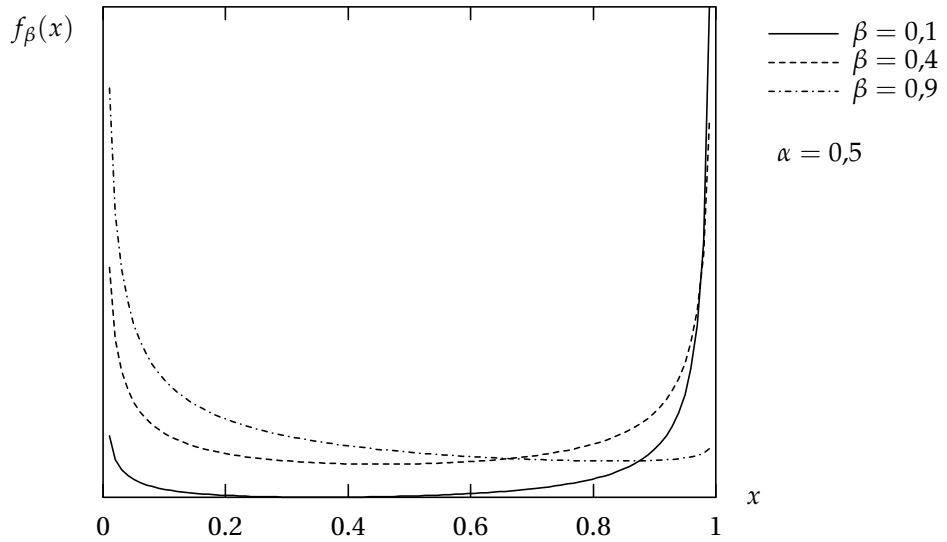


Figura C.3.: Diferentes formas de la distribución beta, cuando ambos parámetros son menores de 1

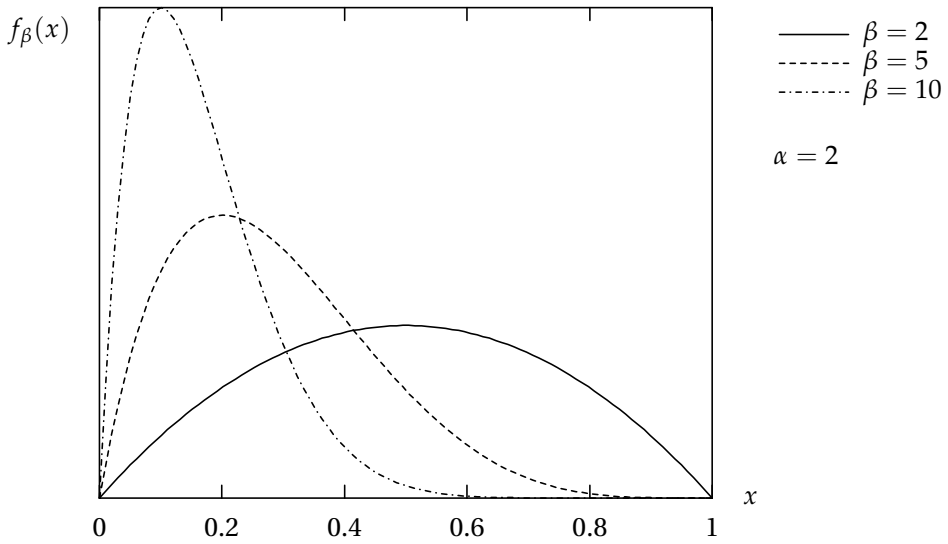


Figura C.4.: Diferentes formas de la distribución beta, cuando ambos parámetros son mayores de 1

- La varianza toma el valor $\dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

C.4. Aplicación de la función beta para modelar el tiempo de ejecución de un trabajo

Una variable aleatoria $\mathcal{X}$ que siga la distribución beta, puede tomar cualquier valor real entre 0 y 1. En cambio un tiempo de ejecución de una tarea es una variable aleatoria discreta que toma valores entre los enteros $C^{\min}$ y $C^{\max}$.

El primer paso es, dados un par de enteros a y b, obtener otra variable aleatoria, $\mathcal{X}'$, cuya distribución tiene la misma forma que la de $\mathcal{X}$, pero escalada y desplazada de modo que $\mathcal{X}'$ toma valores reales en el intervalo $[a, b]$. Para ello basta definir $\mathcal{X}$ en la forma siguiente:

$$\mathcal{X}' = a + \mathcal{X}(b-a)$$

Si $\mathcal{X}$ sigue una distribución beta de parámetros α y β, entonces la función de densidad de $\mathcal{X}'$ vendrá dada por:

$$f_{\mathcal{X}'}(x) = \frac{(x-a)^{\alpha-1}(b-x)^{\beta-1}}{B(\alpha,\beta)(b-a)^{\alpha+\beta-1}} \tag{C.3}$$

donde $B(\alpha,\beta)$ es la función beta definida en la ecuación (C.1).

Si se conocen los valores máximo y mínimo $[a,b]$ que puede tomar $\mathcal{X}'$, su valor medio $\bar{X}'$ y su valor más probable X'_M, es posible estimar los valores de los parámetros α y β con las siguientes ecuaciones:

$$\hat{\alpha} = \frac{(\bar{X}' - a)(2X'_M - a - b)}{(X'_M - \bar{X}')(b - a)} \tag{C.4}$$

$$\hat{\beta} = \frac{(b - \bar{X}')\hat{\alpha}}{(\bar{X}' - a)} \tag{C.5}$$

El segundo paso es muestrear la función continua para obtener una PF para una variable discreta $\mathcal{X}''$. La forma correcta de hacerlo, que garantiza que la $\mathcal{X}''$ obtenida es más pesimista que $\mathcal{X}'$, sería muestrear su función de distribución, que es la integral de la función de densidad, para así obtener la función de probabilidad acumulada de $\mathcal{X}''$. Es decir:

$$F_{\mathcal{X}''}(v) = \int_{-\infty}^{\lfloor v \rfloor} f_{\mathcal{X}'}(x)dx \qquad v \in [a,b] \tag{C.6}$$

y a partir de ella obtener la función de probabilidad por diferenciación, es decir:

$$f_{\mathcal{X}''}(i) = F_{\mathcal{X}''}(i) - F_{\mathcal{X}''}(i-1) \qquad i = a, a+1, \ldots, b-1, b \tag{C.7}$$

Una aproximación razonable mucho más sencilla de calcular consiste simplemente en evaluar $f_{\mathcal{X}'}(i)$ para cada posible i entero en $[a,b]$ y posteriormente normalizar las muestras para que su suma sea 1. Es decir:

$$f_{\mathcal{X}''}(i) \approx \frac{f_{\mathcal{X}'}(i)}{\sum_{j=a}^{b} f_{\mathcal{X}'}(j)} \qquad i = a, a+1, \ldots, b-1, b \tag{C.8}$$

La ventaja de este método aproximado es que es más rápido de computar, ya que no es necesario resolver la integral (que no tiene forma cerrada en general y debe calcularse por métodos numéricos). Por otro lado, esta aproximación produce una PF cuya forma es más "fiel" a la de la distribución beta que se ha muestreado.

Por el contrario, el método basado en la integral produce una PF cuya forma se aleja bastante de la beta muestreada (tanto más cuanto menor sea el número de puntos de muestreo) y su principal ventaja consiste en que se garantiza que la PF obtenida es más pesimista que la beta muestreada.

No obstante, en el contexto de esta tesis, la beta muestreada se utiliza tan sólo como una forma de generar PFs arbitrarias de cara a medir el coste de computación, mediante generación automática de sistemas sintéticos. En este caso no es importante garantizar el pesimismo de la distribución, sino que solo interesa el

número de puntos que la componen y su forma general. Por ello la aproximación directa es perfectamente válida.

La figura C.5 muestra el resultado de aplicar el muestreo directo, o el pesimista basado en la integral, a una función beta de parámetros $\alpha = 2$, $\beta = 4$. En la representación de la función de probabilidad puede verse que el muestreo directo sigue perfectamente la forma de la función de densidad original, mientras que el pesimista va a veces por debajo y otras veces por encima. En la figura C.6 se muestra la función de probabilidad acumulada, y se comprueba cómo la versión pesimista siempre está por debajo de la distribución beta que se muestrea, mientras que la versión por muestreo directo no.

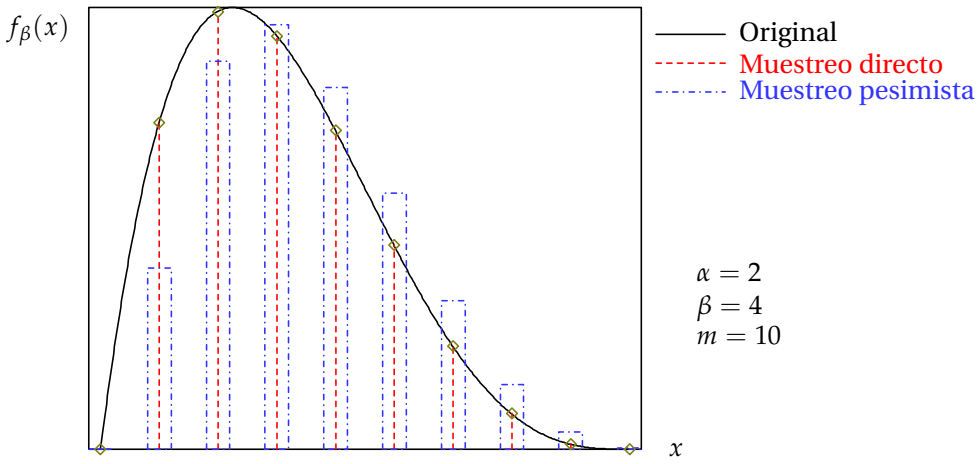


Figura C.5.: Comparación de las PF obtenidas por muestreo directo y por muestreo pesimista, con la función de densidad de la beta

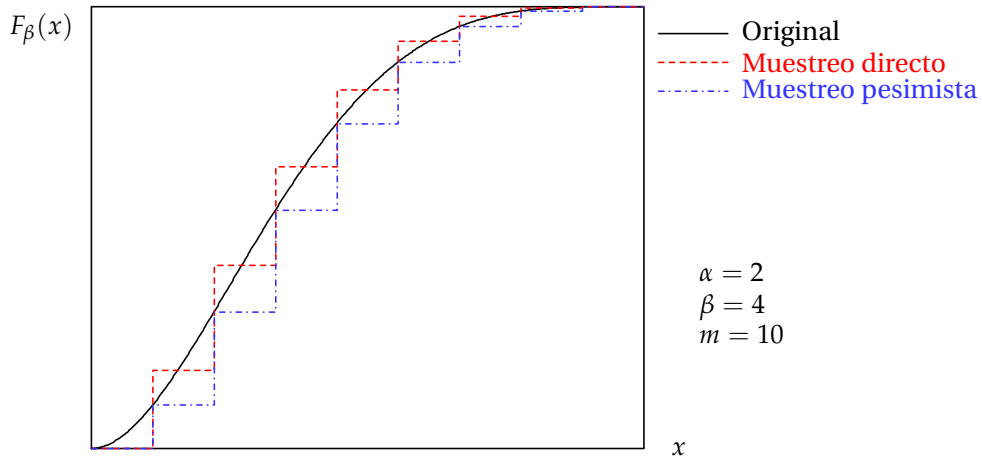


Figura C.6.: Comparación de las CDF (probabilidad acumulada) obtenidas por muestreo directo y por muestreo pesimista, con la función de distribución de la beta

Bibliografía

L. Abeni y G. Buttazzo (1998). Integrating Multimedia Applications in Hard Real-Time Systems. En *Proceedings of the IEEE Real-Time Systems Symposium*. Madrid, Spain.

L. Abeni y G. Buttazzo (1999). QoS Guarantee Using Probabilistic Deadlines. En *Proc. of the Euromicro Conference on Real-Time Systems*. York, UK.

L. Abeni y G. Buttazzo (2001). Stochastic Analysis of a Reservation Based System. En *Proc. of the 9th International Workshop on Parallel and Distributed Real-Time Systems*. San Francisco.

A. K. Atlas y A. Bestavros (1998). Statistical Rate Monotonic Scheduling. En *Proc. of the 19th IEEE Real-Time Systems Symposium*, págs. 123–132.

N. C. Audsley y A. Burns (1990). Real-time System Scheduling. Inf. Téc. YCS 134, University of York.

T. P. Baker (1991). Stack-Based Scheduling of Realtime Processes. *Real-Time Systems*, **3**(1).

G. Bernat, A. Colin y S. Petters (2002). WCET Analysis of Probabilistic Hard Real-Time Systems. En *Proc. of the 23rd IEEE Real-Time Systems Symposium*.

A. Burns y A. Wellings (2001). *Real-Time System and Programming Languages*. Addison Wesley.

G. Buttazzo (1997). *Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers.

G. C. Canavos (1988). *Probabilidad y Estadística. Aplicaciones y Métodos*. McGraw Hill. ISBN 968-451-856-0.

J. L. Díaz, J. M. López y D. F. García (2002). Probabilistic Analysis of the Response Time in a Real-Time System. En *Proc. of the 1st CARTS Workshop on Advanced Real-Time Technologies*. Aranjuez, Spain. Also available as Technical Report at http://www.atc.uniovi.es/research/PART01.pdf.

J. L. Díaz, D. F. García, K. Kim, C.-G. Lee, L. Lo Bello, J. M. López, S. L. Min y O. Mirabella (2002*a*). Stochastic Analysis of Periodic Real-Time Systems. En *Proc. of the 23rd IEEE Real-Time Systems Symposium*, págs. 289–300. Austin, Texas.

J. L. Díaz, J. M. López y D. F. García (2002*b*). Análisis Estocástico del Tiempo de Respuesta. En *Actas de las V Jornadas de Tiempo Real*. Cartagena, España.

G. Fohler (1993). Changing Operational Modes in the Context of Pre Run-Time Scheduling. *IEIC Transactions on Information and Systems, special issue on responsive computer systems*, págs. 1333–1340.

M. K. Gardner (1999). *Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems*. Tesis Doctoral, University of Illinois, Urbana-Champaign.

M. K. Gardner y J. W. Liu (1999). Analyzing Stochastic Fixed-Priority Real-Time Systems. En *Proc. of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.

X. S. Hu y E. H.-M. Zhou, Tao ad Sha (2001). Estimating Probabilistic Timing Performance for Real-Time Embedded Systems. *IEEE Transactions on VLSI Systems*, **9**(6):págs. 833–844.

A. Kalavade y P. Moghê (1998). A Tool for Perfomance Estimation of Networked Embedded End-Systems. En *Proceedings of Design Automation Conference*.

K. Kim, L. L. Bello, S. L. Min y O. Mirabella (2002). On Relaxing Task Isolation in Overrun Handling to Provide Probabilistic Guarantees to Soft Real-Time Tasks with Varying Execution Times. En *Proc. of the 14th Euromicro Conference on Real-Time Systems*.

J. P. Lehoczky (1990). Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. En *Proc. of the 11th IEEE Real-Time Systems Symposium*, págs. 201–209.

J. P. Lehoczky (1996). Real-Time Queueing Theory. En *Proc. of the 17th IEEE Real-Time Systems Symposium*, págs. 186–195.

J. P. Lehoczky (1997). Real-Time Queueing Network Theory. En *Proc. of the 18th IEEE Real-Time Systems Symposium*, págs. 58–67.

J. P. Lehoczky, L. Sha y Y. Ding (1989). The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. En *Proc. of the 10th IEEE Real-Time Systems Symposium*.

J. Leung y J. Whitehead (1982). On the Complexity of Fixed Priority Scheduling of Periodic Real-Time Tasks. *Performance Evaluation*, **2**(4):págs. 237–250.

C. L. Liu y J. Layland (1973). Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, **20**(1):págs. 46–71.

S. Manolache (2002). *Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times*. Tesis Doctoral, Linköping University.

S. Manolache, P. Eles y Z. Peng (2001). Memory and Time-Efficient Schedulability Analysis of Task Sets with Stochastic Execution Times. En *Proc. of the 13th Euromicro Conference on Real-Time Systems*, págs. 19–26.

M. Matsumoto y T. Nishimura (1998). Mersene Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator. *ACM Transactions on Modeling and Computer Simulation*, **8**(1):págs. 3–30.

S. P. Meyn y R. Tweedie (1993). *Markov Chains and Stochastic Stability*. Control and Communication Engineering Series. Springer Verlag, London.

A. K. Mok y D. Chen (1997). A Multiframe Model for Real-Time Tasks. *IEEE Transactions on Software Engineering*, **23**(10):págs. 635–645.

D.-T. Peng y K. G. Shin (1993). Optimal Scheduling of Cooperative Tasks in a Distributed System Using an Enumerative Method. *IEEE Transactions on Software Engineering*, **19**(3):págs. 253–267.

W. H. Press, W. T. Vettering, S. A. Teukolsky y B. P. Flannery (1992). *Numerical Recipes in C*. Cambridge University Press, 2ª ed$^{\text{ón}}$.

W. Rudin (1966). *Real and Complex Analysis*. McGraw-Hill, New York, NY.

L. Sha, R. Rajkumar y J. P. Lehoczky (1990). Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, **39**(9):págs. 1175–1185.

K. G. Shin y P. Ramanathan (1994). Real-Time Computing: A New Discipline of Computer Science and Engineering. *Proceedings of the IEEE*, **82**(1):págs. 6–24.

J. A. Stankovic (1988). Misconceptions about Real-Time Computing: A Serious Problem for the Next Generation Systems. *IEEE Computer*, **21**(10):págs. 10–19.

J. A. Stankovic (1996). Real-Time and Embedded Systems. *ACM Computing Surveys*, **28**(1):págs. 205–208.

J. A. Stankovic y K. Ramamritham (1990). Editorial: What is Predictability for Real-Time Systems? *The Journal of Real-Time Systems*, **2**:págs. 247–254.

J. Sun (1997). *Fixed-Priority End-to-End Scheduling in Distributed Real-Time Systems*. Tesis Doctoral, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu y J.-S. Liu (1995). Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times. En *Proc. of the Real-Time Technology and Applications Symposium*, págs. 164–173. Chicago, Illinois.

K. Tindell y J. Clark (1994). Holistic Schedulability Analysis for Distributed Real-Time Systems. *Microprocessing and Microprogramming*, **50**(2–3):págs. 117–134.

K. Tindell, A. Burns y A. J. Wellings (1994). An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Real-Time Systems*, **6**:págs. 133–151.

R. Tweedie (2000). Markov Chains: Structure and Applications. En D. Shanbhag y C. Rao, eds., *Handbook of Statistics*, tomo 19, págs. 817–851. Elsevier Amsterdam.

C. Weems y S. Dropsho (1995). Real-Time Computing: Implications for General Microprocessors. Inf. Téc. UM-CS-1995-042, University of Massachusetts.

J. Xu y D. L. Parnas (1990). Scheduling Processes with Release Times, Deadlines, Precendence and Exclusion Relations. *IEEE Transactions on Software Engineering*, págs. 360–369.

T. Zhou, X. S. Hu y E. H.-M. Sha (1999). A Probabilistic Performance Metric for Real-Time System Design. En *Proc. of the Hardware/Software Co-Design Symposium*, págs. 90–94.