



FACULTADE DE INFORMÁTICA

Departamento de matemáticas

PROXECTO FIN DE CARREIRA DE ENXEÑERÍA INFORMÁTICA

Sistema de ayuda a la toma de decisión para inversiones en redes de telecomunicaciones y telefonía móvil bajo incertidumbre en la demanda

Autor: Alberto Bengoa Moreno

Director: Carlos Vázquez Cendón

A Coruña, 20 de Xuño, 2005

Título del proyecto: **Sistema de ayuda a la toma de decisión para inversiones en redes de telecomunicaciones y telefonía móvil bajo incertidumbre en la demanda**

Clase de proyecto: **Investigación y Desarrollo**

Autor: **Alberto Bengoa Moreno**

Director de proyecto: **Carlos Vázquez Cendón**

Fecha de lectura y defensa:

Miembros del tribunal:

- Presidente:

- Vocal 1º:

- Vocal 2º:

- Vocal 3º:

- Secretario:

Calificación obtenida:

A mi familia, que me ha soportado todo el tiempo que ha durado todo el proyecto, y me ha apoyado sin desfallecer.

A mis amigos, que han acabado sabiendo tanto como yo de valoración de redes y de métodos numéricos.

Al Departamento de Matemáticas y, en especial, a Carlos Vázquez Cendón por toda la ayuda prestada, sin la que este proyecto nunca se habría acabado.

Resumen del proyecto:

En el ámbito actual de los proveedores de acceso a Internet y las operadoras de telefonía móvil el problema de la dimensión de las infraestructuras es de una importancia capital; las inversiones en infraestructuras de comunicación se deben realizar de tal forma que la red tenga suficiente capacidad para proveer servicio al nivel demandado pero sin estar sobredimensionada, para no desperdiciar capital en infraestructuras no utilizadas. El presente Proyecto Fin de Carrera propone un sistema de ayuda a la decisión que asesora a estas empresas en la cuantía y momento de las inversiones en sus infraestructuras de forma que maximicen el beneficio obtenido.

Para alcanzar el objetivo propuesto, el sistema aplica técnicas financieras modernas en oposición a las técnicas clásicas de valoración de infraestructuras, obteniendo decisiones mejores que las obtenidas por estos métodos tradicionales. Las técnicas financieras aplicadas involucran el uso de métodos numéricos en los cálculos necesarios; estos métodos numéricos serán desarrollados en detalle.

El uso de métodos numéricos normalmente conlleva asociada una alta carga computacional, de forma que se requiere hardware de altas prestaciones para la resolución del problema en tiempos reducidos. Para reducir este tiempo, se estudia la resolución del problema de forma distribuida, empleando para ello una red de ordenadores de propósito general como los que pueden hallar en una empresa típica, resultando en un tiempo de cálculo menor, con una inversión de capital muy inferior a un sistema hardware especializado o de altas prestaciones.

El sistema resultante será fácil de utilizar por un usuario que no sea experto en informática, y los resultados se presentarán de forma fácilmente legible. Para cumplir estos objetivos, se desarrollará una interfaz gráfica de usuario que permita la interacción con el sistema de forma agradable. De esta forma, se contará con una herramienta sencilla de utilizar que permita apoyar las decisiones empresariales para maximizar el beneficio que se obtenga de las infraestructuras.

Lista de palabras clave:

Redes de fibra óptica, redes de telefonía móvil, métodos numéricos, método de elementos finitos, método de características, cálculo distribuído, valoración de inversiones, Java.

Índice general

I	Introducción	21
1.	Introducción	23
II	Aspectos teóricos	27
2.	Modelo matemático para redes de acceso a Internet	29
2.1.	Modelo matemático	29
2.1.1.	Función de beneficio	31
2.1.2.	Mantenimiento de la línea	33
2.1.3.	Toma de decisiones	34
3.	Modelo matemático para redes de telefonía móvil	43
3.1.	Introducción	43
3.2.	Modelo matemático	43
3.2.1.	Función de beneficio	45
3.2.2.	Mantenimiento del clúster	46
3.2.3.	Toma de decisiones	46
4.	Derivación del método de elementos finitos	59
4.1.	Reescalado de la ecuación	59
4.2.	Escritura en forma conservativa	60
4.3.	Condiciones de contorno	61
4.4.	Discretización en tiempo: método de las características	61
4.4.1.	Discretización por elementos finitos	64
5.	Resolución distribuida del problema	67
5.1.	Resolución de una EDP general	67
5.2.	Redes de fibra óptica	69

5.3. Redes de telefonía móvil	72
---	----

III Implementacion de la aplicación 75

6. Desarrollo de la aplicación 77

6.1. Método de elementos finitos	77
6.1.1. Validación del prototipo	77
6.1.2. Optimizaciones aplicadas	80
6.2. Decisiones para redes de acceso a Internet	81
6.3. Decisiones para redes de telefonía móvil	89
6.4. Sistema de cálculo distribuído	91
6.4.1. Requisitos del sistema de cálculo distribuído	91
6.4.2. Diseño del sistema	92
6.4.3. Utilización del sistema	107
6.5. Aplicación final	114
6.5.1. Análisis de requisitos de la aplicación	114
6.5.2. Diseño de la aplicación	115

7. Resultados obtenidos 145

7.1. Resultados numéricos	145
7.2. Resultados para decisiones sobre redes de fibra óptica	149
7.2.1. Parámetros empleados	149
7.2.2. Resultados obtenidos	149
7.3. Resultados para decisiones sobre redes de telefonía móvil	152
7.3.1. Parámetros empleados	152
7.3.2. Resultados obtenidos	152
7.4. Resultados del cálculo distribuído	155
7.4.1. Resultados del sistema de paralelización	156
7.4.2. Valoración de redes con cálculo distribuído	163

IV Conclusiones 173

8. Conclusiones y trabajos futuros 175

8.1. Conclusiones	175
8.2. Líneas de trabajo futuras	177

<i>ÍNDICE GENERAL</i>	13
A. Planificación del proyecto	181
A.1. Definición de las actividades	181
A.2. Restricciones entre actividades	182
A.3. Planificaciones	182

Índice de figuras

2.1. Distribución de los tiempos referentes a las decisiones	35
2.2. División del tiempo en subintervalos	36
2.3. Superposición de las zonas de tipo II	36
2.4. Distribución de los tiempos relevantes del problema	37
2.5. Cálculo del valor de una línea entre t_1 y t_2 considerando el mantenimiento de la misma	38
2.6. Cálculo del valor de las n líneas en una zona de tipo I	39
2.7. Esquema del algoritmo de decisión de mejora para la línea i	41
3.1. Efecto del factor ϕ en la función de renta	46
3.2. Superposición de varias zonas de tipo II cuando γ es mayor que Δt_n	48
3.3. Evolución del valor de l para una mejora	49
3.4. Diferentes valores de l para diferentes mejoras superpuestas	50
3.5. Distribución de la información en el problema	51
3.6. Cálculo del valor de los n clústers en una zona de tipo I	53
3.7. El flujo de información para decisiones parcialmente completadas	54
3.8. El flujo de información para nuevas decisiones (mejoras completadas)	55
3.9. El flujo de información para decisiones en t_n	57
4.1. Proceso de resolución de la ecuación en derivadas parciales	65
5.1. Dependencias entre tareas del algoritmo de redes de fibra óptica	70
5.2. Posible escenario de uso excesivo de memoria	71
6.1. Aparición de discontinuidades	79
6.2. Forma de la solución V^n	82
6.3. Resolución del conjunto de ecuaciones con varios dominios	83
6.4. Comparación entre las mallas de cada una de las líneas	83
6.5. Necesidad de interpolación cuando se utilizan varias mallas	84

6.6. Una única malla para la resolución de las diferentes ecuaciones	85
6.7. El valor de ambas líneas en la fecha final t_{u_2}	86
6.8. El valor de ambas líneas en la fecha de notificación t_n	86
6.9. El valor de ambas líneas en la fecha de mejora t_{u_1}	87
6.10. Casos de uso para el sistema de cálculo distribuido	92
6.11. La especificación de un nodo de cálculo	95
6.12. Clases para la especificación de un problema	95
6.13. Clases de la implementación de un nodo de cálculo	97
6.14. Especificación e implementación del nodo central	99
6.15. Clases de la fachada cliente del sistema	102
6.16. Una secuencia que produce una excepción <code>PermissionDeniedException</code>	104
6.17. Secuencia de arranque de un nodo de cálculo	105
6.18. Finalización normal de un nodo	105
6.19. Eliminación de un nodo cuando deja de ser accesible	106
6.20. Proceso de desactivación y reactivación de un nodo.	106
6.21. Obtención de un nodo por el programa cliente	107
6.22. Secuencia para la resolución de un problema de forma remota	108
6.23. Interfaz gráfica para el nodo central	109
6.24. Interfaz gráfica para un nodo de cálculo	110
6.25. Casos de uso de la aplicación	115
6.26. Clases para la especificación de un problema	117
6.27. Clases adicionales para la definición de un módulo de resolución de un problema	118
6.28. Clase que administra los problemas instalados en la aplicación	120
6.29. Las clases que definen la interfaz gráfica común	121
6.30. La interfaz gráfica de la aplicación, sin ningún problema abierto.	121
6.31. La secuencia de creación del cuadro de diálogo para un nuevo problema	122
6.32. El cuadro de diálogo para crear un problema nuevo	122
6.33. Una instancia de <code>ProblemFrame</code>	123
6.34. La secuencia de creación de una ventana de problema	123
6.35. El cuadro de diálogo para selección de fichero	124
6.36. La aplicación con dos problemas abiertos	125
6.37. La secuencia de resolución de un problema	126
6.38. El indicador de progreso de las ventanas de problema	126
6.39. Las clases que resuelven la ecuación mediante el método de elementos finitos con características	127

6.40. El motor de resolución de problemas de fibra óptica	129
6.41. Clases requeridas en la resolución distribuída del problema	131
6.42. Clases para la interfaz del motor de cálculo del módulo de fibra óptica	131
6.43. Las clases de entrada de datos para la valoración de redes de fibra óptica	132
6.44. El aspecto del componente de entrada de datos	133
6.45. Página de datos de frontera	133
6.46. Página de datos de la malla	133
6.47. Página de datos financieros del problema	134
6.48. Página de datos de las diferentes líneas	134
6.49. Página de datos de mejora de líneas	135
6.50. Cuadro de diálogo cuando existen datos modificados en un cambio de página	135
6.51. Las clases de salida de datos para la valoración de redes de fibra óptica	136
6.52. El aspecto del componente de la solución del problema	136
6.53. Clases del motor de resolución para redes de telefonía móvil	138
6.54. Clases para la resolución dsitribuída del problema	139
6.55. Las clases de la interfaz del módulo para redes de telefonía móvil	140
6.56. El componente de entrada de datos para redes de telefonía móvil	140
6.57. La estructura de clases del componente de entrada de datos	141
6.58. Página para los datos de frontera	141
6.59. Página para los datos de la malla	142
6.60. Página para los datos financieros	142
6.61. Página para los datos de los diferentes clústers	143
6.62. Página para los datos de mejora de los clústers	143
6.63. Componente Swing que muestra los resultados del problema	143
6.64. Clases del componente de salida de datos	144
7.1. Soluciones obtenidas con diferentes mallas	146
7.2. Soluciones obtenidas con diferentes mallas (detalle)	146
7.3. Convergencia de la solución	148
7.4. Convergencia de la solución (detalle)	148
7.5. Tiempos de ejecución secuencial para distintos valores de <i>max</i>	158
7.6. <i>Speedup</i> obtenido utilizando un nodo	159
7.7. <i>Speedup</i> obtenido utilizando dos nodos	161
7.8. <i>Speedup</i> obtenido utilizando cuatro nodos	162
7.9. <i>Speedup</i> obtenido utilizando seis nodos	164
7.10. <i>Speedup</i> para el mejor valor de <i>max</i> (10000000)	165

7.11. <i>Speedup</i> para el peor valor de <i>max</i> (500000)	166
7.12. Evolución del mejor <i>speedup</i> según el número de nodos de cálculo	166
7.13. <i>Speedup</i> obtenido en el problema de toma de decisiones para redes de acceso a Internet . .	169
7.14. <i>Speedup</i> obtenido en el problema de toma de decisiones para redes de telefonía móvil . . .	171
A.1. Las restricciones entre las actividades del proyecto	183
A.2. La planificación del proyecto	183
A.3. Una posible planificación del proyecto	184
A.4. La mejor planificación temporal del proyecto	184

Índice de cuadros

7.1. Número de puntos de cada malla	149
7.2. Datos de entrada para el problema de las redes de acceso a Internet	150
7.3. Datos de las líneas usadas	150
7.4. Costes de mejora de las líneas	150
7.5. Toma de decisiones para los dos primeros años del caso base	150
7.6. Efecto de la congestión de las líneas en las decisiones	152
7.7. Datos de entrada para el problema de las redes de telefonía móvil	152
7.8. Datos de los clústers usados	153
7.9. Costes de mejora de los clústers	153
7.10. Datos numéricos usados	153
7.11. Resultados para los diferentes niveles de refinamiento de malla	154
7.12. Resultados para diferentes valores de k	154
7.13. Influencia del factor de seguridad Φ	155
7.14. Tiempos de ejecución secuencial para distintos valores de max	157
7.15. Speedup obtenido utilizando un nodo de cálculo	160
7.16. Speedup obtenido utilizando dos nodos de cálculo	160
7.17. Speedup obtenido utilizando cuatro nodos de cálculo	161
7.18. Speedup obtenido utilizando seis nodos de cálculo	163
7.19. Tiempos obtenidos en el problema de toma de decisiones para redes de acceso a Internet	168
7.20. Tiempos obtenidos en el problema de toma de decisiones para redes de telefonía móvil	171

Parte I

Introducción

Capítulo 1

Introducción

En el ámbito empresarial es de vital importancia las decisiones sobre inversiones de capital que se acometerán por la empresa. Efectivamente, una mala política a la hora de invertir puede conducir con facilidad a un desaprovechamiento de los recursos implicados, produciéndose en último término pérdidas de beneficios, que llevadas al extremo pueden hacer que la empresa quiebre. Es por ello que las inversiones se realizan en muchas ocasiones con la aprobación de asesores financieros, que ayudan a realizar los desembolsos con unas ciertas garantías de que no son contraproducentes.

En el contexto de las comunicaciones actuales, esta situación es muy común. Por ejemplo, un proveedor de acceso a Internet posee una infraestructura de fibra óptica con la cual da acceso a la red Internet a sus clientes. Estas redes de fibra óptica determinan a cuántos usuarios se les puede dar servicio, y con qué nivel de calidad. Así, el problema principal con el que se enfrenta un proveedor de acceso a Internet, en cuanto a inversiones de capital, es el del *dimensionamiento óptimo de la red de comunicación*, es decir, la capacidad que deben tener las líneas con las que ofrece acceso a sus clientes. Con unas redes de baja capacidad, se podrá dar servicio a pocos usuarios (o a muchos usuarios con baja calidad de servicio), mientras que con unas redes de mayor capacidad se puede atender a un mayor número de clientes. Estas consideraciones, junto con los precios de las diferentes líneas de fibra óptica, determinarán qué decisiones se toman al invertir capital en la fibra óptica. Históricamente, se ampliaba la capacidad de una línea cuando llegaba a un porcentaje de utilización del 50 %, sin tener en cuenta otras consideraciones, produciéndose por norma general un sobredimensionamiento de la infraestructura de fibra óptica, y una pérdida de beneficios apreciable.

Una situación parecida es la que tiene que afrontar un operador de telefonía móvil. El servicio que ofrece a sus clientes está sustentado en *clústers* que dan cobertura a una zona; estas estaciones tienen una capacidad de atender usuarios, con un máximo de usuarios simultáneos. La capacidad de un clúster viene dada por dos factores: el número de estaciones base que tiene y el número de portadores que hay en cada estación. Las estaciones base cubren una determinada zona cada una, y están distribuidas de forma que

cubran un área extensa; los portadores le dan capacidad de atender llamadas a las estaciones base. Así, si un cliente intenta utilizar un clúster que ya está completamente utilizado, su llamada se bloqueará, y tendrá que volver a intentarlo más adelante. La decisión que tiene que tomar el operador de telefonía es la del dimensionamiento de dichos clústers, de forma que invierta el menor capital posible maximizando sus beneficios.

En ambas situaciones existe incertidumbre en la demanda del servicio y será preciso modelar matemáticamente dicha incertidumbre. Una posibilidad es utilizar ecuaciones estocásticas similares a las que gobiernan los precios de activos financieros. De este modo, a estas dos situaciones se les puede aplicar un método desarrollado en el contexto de las opciones financieras para conseguir tomar decisiones más inteligentes que simplemente mejorar la infraestructura cuando llegue a un 50 % de utilización. Para aplicarlas, se considera la *demandas de capacidad* como una variable aleatoria que será el principal factor a tener en cuenta a la hora de tomar las decisiones. Esta demanda de capacidad es difícil de predecir, por lo que es el principal factor de riesgo para una de las empresas que se han descrito anteriormente. Situaciones en las que las infraestructuras están bien dimensionadas (que han seguido una política de inversiones acertada) se dan con una demanda baja y unas infraestructuras de baja capacidad o con una demanda alta y unas infraestructuras de alta capacidad. Una mala previsión de la demanda que la empresa va a tener conduce fácilmente a situaciones indeseables, que se pueden dar de dos formas:

- En primer lugar, situaciones con alta demanda e infraestructuras de baja capacidad, que harán que la empresa pierda muchos clientes debido a la poca calidad de su servicio, con la consecuente pérdida de los beneficios que esos clientes producen,
- En segundo lugar, situaciones en las que una infraestructura de alta capacidad tiene muy baja demanda. En esta situación, es acertado suponer que la empresa ha realizado unas inversiones de capital en infraestructuras muy elevadas que no se van a amortizar de la forma prevista, ya que no hay suficientes clientes para obtener el beneficio esperado.

Ambas situaciones son poco deseables por la empresa, y deben ser evitadas llevando a cabo una inteligente política de inversiones. Sin embargo, esto puede ser difícil, ya que en los dos contextos expuestos la demanda sigue pautas cada vez más aleatorias. En el ámbito del acceso a Internet la forma de los contratos con el cliente está moviéndose lentamente hacia una nueva posición. En el pasado, la mayor parte de los contratos con clientes de acceso a Internet eran de larga duración, pero cada vez más las compañías dan opción a contratos de alquiler de capacidad de menor duración, haciendo que los clientes contraten el acceso a Internet con mayor precisión, y haciendo, por lo tanto, mucho más difícil el predecir qué demanda tendrá un proveedor en un momento dado. En cuanto a la telefonía móvil, la situación es similar, ya que el tráfico de voz es extremadamente volátil, haciendo que sea muy fácil incurrir en errores de predicción de demanda.

Para mitigar estos problemas, en este Proyecto Fin de Carrera se propone un sistema que analice la situación actual de la empresa y proponga las mejoras en infraestructura que la empresa debe llevar a cabo y en qué momento debe realizarlas, dependiendo de la demanda que tenga por parte de los clientes. Por ejemplo, las decisiones que propondrán serán del tipo *mejora la línea OC-12 a una OC-48 dentro de un mes si la demanda que tienes en ese momento supera el 80 %*. Estas decisiones se tomarán en base al beneficio percibido por el poseedor de la infraestructura de comunicación, de forma que ese beneficio sea máximo (que claramente es lo que le interesa a la empresa). Así se tendrá una guía que permita tomar las decisiones con ciertas garantías de que esas decisiones conducirá a un dimensionamiento correcto de las redes, y por tanto, se evitarán las situaciones de pérdida de beneficios que se han descrito anteriormente.

Para resolver el problema propuesto es preciso plantear modelos adecuados para valorar las inversiones. Dichos modelos se basan en la metodología de cobertura dinámica y las opciones de inversión están gobernadas por ecuaciones en derivadas parciales. La toma de decisiones requiere técnicas numéricas de resolución de las mismas. Por otra parte, la estructura del problema de toma de decisiones indica que puede ser factible la aplicación de computación distribuida que mitigue el elevado coste computacional.

La presente Memoria está redactada según la siguiente estructura:

Parte I (Introducción) contiene esta introducción.

Parte II (Aspectos teóricos) : desarrolla los modelos teóricos utilizados en la aplicación, en cuatro capítulos:

- Modelo matemático para redes de acceso a Internet: explica el modelo que se sigue para la toma de decisiones para redes de fibra óptica.
- Modelo matemático para redes de telefonía móvil: presenta el modelo teórico involucrado en las decisiones de ampliación de redes de telefonía celular.
- Derivación del método de elementos finitos: muestra la derivación del método de elementos finitos con características que se emplea en la resolución de los modelos anteriores.
- Resolución distribuida del problema: presenta un estudio sobre la viabilidad de la resolución distribuida del cálculo de las tomas de decisión y la forma en la que debería aplicarse dicha resolución distribuida.

Parte III (Implementación de la aplicación) : contiene dos capítulos, ambos referidas a los aspectos prácticos de la implementación del sistema:

- Implementación de la aplicación: expone la construcción de la aplicación resultante, centrándose en el diseño y codificación de la misma.
- Resultados obtenidos: presenta los resultados que se han obtenido en el conjunto del Proyecto Fin de Carrera, organizados en áreas de interés.

Parte IV (Conclusiones) : resume las conclusiones obtenidas en el transcurso del proyecto, y propone líneas futuras de trabajo.

Parte II

Aspectos teóricos

Capítulo 2

Modelo matemático para redes de acceso a Internet

En el presente capítulo se desarrolla el modelo que se emplea para la resolución del problema de la toma de decisiones en el ámbito de las líneas de comunicación de un proveedor de acceso a Internet, que es el primero de los problemas que se van a resolver en este proyecto. El modelo propuesto contiene dos partes, la primera de ellas referida a la valoración de una línea de fibra óptica, y la segunda de ellas referida a las tomas de decisión de mejora de una línea a un tipo de mayor capacidad. El modelo ha sido extraído de [4] y ha sido adaptado para su integración en la aplicación que se va a desarrollar.

2.1. Modelo matemático

En el problema de la valoración de inversiones de redes de telecomunicaciones se va a partir del concepto de *demanda de ancho de banda*. En efecto, para un proveedor de acceso a Internet esta es una variable muy importante, ya que determina la cantidad de beneficio que va a obtener y el tamaño (capacidad) que deben tener sus infraestructuras. Así, la demanda de ancho de banda va a considerarse como la variable principal a la hora de realizar mejoras en las redes del proveedor de acceso a Internet; este enfoque es contrario a la tradicional situación financiera, en la cual en el mercado el beneficio que se obtiene en una transacción depende del precio de lo que se está comerciando. En efecto, en el ámbito de las redes de telecomunicaciones el principal factor en el beneficio percibido por un proveedor de acceso a Internet es *la cantidad de ancho de banda* que puede vender, y no el precio del Megabit/segundo que ofrece a los clientes. Esto es aún más relevante en el mercado actual, en el que se pueden contratar servicios de acceso por un intervalo de tiempo muy pequeño, incluso horas, en contraposición a los tradicionales contratos de larga duración con un proveedor. De esta forma, el conocimiento acerca de la demanda que va a tener la compañía es de vital importancia a la hora de establecer las estrategias de mercado, ya que

al no estar basada en contratos asegurados por largos períodos de tiempo, la demanda de ancho de banda de una compañía será muy volátil. Una variación imprevista o una estrategia incorrecta pueden hacer que el proveedor de acceso a Internet pierda grandes cantidades de dinero.

En concreto, el modelo que se va a emplear tiene como única variable el ancho de banda que se puede vender, y no el precio al que se vende el Megabit/segundo. Esto permite construir un modelo más sencillo y aplicable que si se tomaran ambos factores como variables del modelo; consecuentemente, los resultados que proporcione este modelo serán menos precisos que los proporcionados por modelos de dos o más factores. Sin embargo, debido a que estudios previos muestran que el precio del ancho de banda se puede predecir con cierta exactitud (este precio sigue una tendencia descendente muy fiable), este parámetro se fija como conocido para utilizar un modelo de un único factor, la demanda de ancho de banda, manteniéndose la fiabilidad de los resultados.

Así, se va a suponer que la demanda de ancho de banda es una variable estocástica. Los estudios realizados demuestran que es una suposición razonable, ya que, al contrario que el precio del Megabit/segundo, el ancho de banda demandado es mucho más difícil de predecir. De este modo, la demanda de capacidad (ancho de banda) será una proceso estocástico Q que cumplirá la ecuación diferencial estocástica [2]:

$$dQ = \mu Q dt + \sigma Q dZ$$

donde μ es la tasa de crecimiento, σ es la volatilidad de la demanda y dZ es el incremento del proceso de Wiener. La variable Q está medida en Megabits.

Basándose en razonamientos de las técnicas cobertura dinámica [14], se puede deducir una ecuación en derivadas parciales de la expresión anterior. Dicha ecuación es la que gobierna el valor de una inversión en una línea de comunicación en términos de la demanda y del tiempo:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 Q^2 \frac{\partial^2 V}{\partial Q^2} + (\mu - k\sigma)Q \frac{\partial V}{\partial Q} - rV + R = 0 \quad (2.1)$$

La función V es el valor de la inversión en la línea de comunicación. Es una función de dos variables ya que depende de la demanda de capacidad (Q) y del tiempo (t). Es decir, $V = V(Q, t)$. El valor de la inversión viene expresado en unidades monetarias. El parámetro r es la tasa de interés libre de riesgo que se aplicará a las transacciones monetarias (que es una medida adimensional), σ y μ son la volatilidad y la tasa de crecimiento de la demanda respectivamente, y R es la función de beneficio, que determina las retribuciones que la línea otorga al poseedor en cada unidad de tiempo (y por lo tanto, viene expresada en unidades monetarias/tiempo). Esta es una función que depende tanto del tiempo como de la demanda de ancho de banda, así que su expresión será $R(q, t)$ (y que se abreviará como R). El parámetro k se denomina *precio del mercado del riesgo* y expresa el beneficio que se obtiene por cada unidad de riesgo que se asume en una inversión dependiente de la demanda de capacidad Q . Así, si se asume riesgo en la inversión en el beneficio se obtendrá una cantidad extra determinada por este parámetro k .

Como en el contexto de las opciones [14], este problema va a ser resuelto hacia atrás en el tiempo. En efecto, en la valoración de opciones se conoce su precio en la fecha de vencimiento, y se quiere conocer su precio en el instante actual. Normalmente, se asume una resolución hacia atrás en el tiempo (aunque se puede hacer también un cambio de variable, de forma que se define un tiempo τ hasta el vencimiento, haciendo que la resolución en τ sea hacia delante; en el presente trabajo, la resolución se hará sobre el tiempo físico t , y por tanto se resolverá hacia atrás. En el problema de la valoración de inversiones se define el *horizonte de inversión* T como el tiempo final del problema, en el que se conoce el valor V (el problema siempre comienza en el tiempo 0 y termina en el tiempo T ; el dominio de la variable t será por tanto $[0, T]$). Así, se puede definir cualquier función f para establecer el valor de la línea en el horizonte de inversión T :

$$V(Q, T) = f(Q) , \quad (2.2)$$

por lo que la función f tiene como dominio el de la demanda de ancho de banda Q . Para cada uno de los posibles valores de q , el valor $f(q)$ establece el valor de la inversión en el punto (q, T) (el punto q en el horizonte de inversión). En los resultados presentados en este trabajo se utilizará la función f nula, es decir:

$$V(Q, T) = f(Q) = 0 \quad (2.3)$$

que especifica que la línea pasados T años deja de tener valor. El horizonte de inversión se tomará de 5 años, de forma que el problema se resolverá entre 0 y 5 años. Aunque es un período corto en el ámbito de las finanzas, en el ámbito de las comunicaciones es razonable suponer una rápida devaluación de las infraestructuras, debido al rápido crecimiento de la capacidad de transmisión y de la demanda de ancho de banda. Así, un margen de 5 años es un intervalo “amplio” si se considera el contexto en el que se está trabajando. De todas formas, tanto el horizonte de inversión como la función f están abiertos a modificaciones, en el caso de que se quieran considerar otros intervalos de tiempo y otros valores en dicho horizonte. Por ejemplo, si se puede sacar algún beneficio extra cuando el hardware se retira o si el retirar las infraestructuras obsoletas requiere un coste adicional; estos supuestos se manejarán con la función f , estableciendo valores positivos o negativos.

En cualquier caso, el modelo del valor de una línea se plantea como una ecuación parabólica retrógrada en derivadas parciales. Se analizan a continuación algunos ingredientes adicionales.

2.1.1. Función de beneficio

La función R , como ya se ha comentado previamente, determina el beneficio que obtiene el poseedor de la línea. Suponiendo que dicha línea tenga una capacidad máxima de transmisión de \bar{Q}_i megabits, entonces la función R tiene como expresión

$$R(Q, t) = \min(Q, \bar{Q}_i) d P_0 e^{(-\alpha t)}, \quad (2.4)$$

donde d es la longitud de la línea, P_0 es el precio de venta actual y α es un factor de decaimiento. Este factor determina la velocidad a la que cae el precio que se cobra por el alquiler de la capacidad de la línea. En esta expresión, P_0 determina el precio que se paga en el momento actual ($t = 0$) por el megabit, luego el término $P_0 e^{(-\alpha t)}$ es el precio del megabit en el momento t , ya que este precio decae con el tiempo (y lo hace, como ya se ha comentado, de una forma bastante predecible). Las unidades en las que se mide P_0 son unidades monetarias/unidades de tiempo/unidades de longitud/megabit (por ejemplo, $\frac{\text{\$}}{\text{año kilometro megabit}})$.

Posibilidades de arbitraje

La función R genera dos cuestiones con respecto a las situaciones de arbitraje. En la terminología financiera, una situación de arbitraje se produce cuando es posible obtener un beneficio inmediato sin ningún tipo de riesgo. En el modelo de inversiones en redes de comunicaciones se pueden presentar posibilidades de arbitraje de dos maneras:

- Al ser el precio $P_0 e^{(-\alpha t)}$ decreciente, se podría producir arbitraje si el ancho de banda no alquilado se pudiese almacenar de alguna forma para posteriormente alquilarlo; sin embargo, en la práctica si en un momento dado una línea sólo está alquilada en un porcentaje inferior al 100 % la capacidad que no se aprovecha se desperdicia de forma irreversible, no siendo posible aprovecharlo de ninguna forma. De hecho, esto hace que el poseedor de la línea de comunicaciones intente siempre maximizar el uso de la línea, para no desperdiciar ancho de banda (que es algo bastante intuitivo).
- Por otra parte, el precio P_0 debe ser siempre el mismo, independientemente de la capacidad de la línea que se estudie. En efecto, si el precio P_0 fuese diferente para cada modelo de línea entonces se podría alquilar capacidad de una línea con un P_0 dado, e inmediatamente venderla como capacidad de otra línea con un P_0 más alto, ganando una cantidad de dinero automáticamente y sin riesgo (posibilidad de arbitraje). Esto añade una restricción al modelo que se está usando: solamente podrá haber un único P_0 , independientemente de la línea que se estudie, para evitar oportunidades de arbitraje.

Congestion de la línea

Una vez descartada la posibilidad de arbitraje, es necesario hacer una reflexión acerca de la función de beneficio R . En efecto, el beneficio es creciente en Q hasta que se alcanza el punto $Q = \bar{Q}_i$, momento en el cual el beneficio se estanca en su valor máximo. Esto quiere decir que el poseedor de la línea va a recibir los mismos beneficios si su línea está utilizada en un 100 % o en un 200 %. Este enfoque no es muy

realista, ya que una línea ocupada al doble de su capacidad producirá un rendimiento muy degradado, y los clientes percibirán un ancho de banda muy inferior al que habían alquilado. De esta forma, los clientes seguramente buscarán otro proveedor de acceso a Internet, de forma que los beneficios se reducirán. Para modelar este efecto de una forma sencilla, se puede cambiar la expresión de R , haciendo que haya un factor de penalización cuando la red alcance una demanda que esté un porcentaje prefijado por encima del 100 %. Por ejemplo, para un margen de un 20

$$R(Q, t) = \begin{cases} \min(Q, \bar{Q}_i) d P_0 e^{(-\alpha t)} & \text{si } Q \leq 1,2\bar{Q}_i \\ 0 & \text{en caso contrario} \end{cases}$$

De esta forma se está penalizando la sobreexplotación de la capacidad de las líneas de comunicaciones. La función definida trata de una forma sencilla la penalización resultante, aunque como se verá en los resultados, es eficaz al penalizar las decisiones en las cuales se supera el valor máximo de capacidad de la línea. Si en algún caso esta función se hace demasiado simple y no captura toda la problemática de una situación en la que una línea se alquile por una mayor capacidad de la que posee, bastará con definir una nueva R que determine de forma correcta el beneficio que se consigue en esas zonas de Q .

2.1.2. Mantenimiento de la línea

Otro factor que se debe incluir en el modelo de las líneas de comunicaciones es el del mantenimiento. En efecto, las infraestructuras necesitan revisiones periódicas y reparaciones que se pueden considerar también periódicas. En todo caso, el mantenimiento se puede considerar como una cantidad que se debe pagar periódicamente (esta suposición es algo muy común en el ámbito empresarial). En el modelo que se está definiendo los pagos de mantenimiento se define un período entre cuotas de mantenimiento consecutivas (Δt_m), y se denota por t^- el momento inmediatamente anterior al pago de un plazo de mantenimiento, y por t^+ el momento inmediatamente posterior al pago de un plazo de mantenimiento. De esta manera, se expresa el pago de una cuota de mantenimiento como

$$V(q, t^-) = V(q, t^+) - M_i d \Delta t_m \quad (2.5)$$

donde M_i es el coste de mantenimiento, expresado en unidades monetarias/unidades de tiempo/unidades de longitud (por ejemplo, $\frac{\text{euros}}{\text{año kilometro}}$). Como en la función de beneficio, d es la longitud de la línea que se está considerando, y Δt_m es el período entre cuotas de mantenimiento consecutivas, medida en unidades temporales (por ejemplo, *años*).

Una consideración que ha de hacerse es que la incorporación de pagos de mantenimiento supone una introducción de discontinuidades en la solución de la ecuación en derivadas parciales en determinados puntos temporales. En ocasiones la aparición de discontinuidades hace que los métodos numéricos realicen aproximaciones mucho peores o incluso incorrectas de la solución de la ecuación, así que es un dato que ha de tenerse en cuenta a la hora de realizar las pruebas del sistema.

2.1.3. Toma de decisiones

Hasta ahora se ha presentado el modelo que gobierna el valor de una línea de comunicaciones en un intervalo temporal $[0, T]$, incluyendo diferentes fórmulas para el modelado de los beneficios obtenidos por el poseedor de la línea, e incluyendo también los pagos que se han de realizar para el mantenimiento de la línea a lo largo del tiempo. Es decir, se tiene modelado el entorno de una línea de comunicación. Sin embargo, el propósito del sistema a construir es ayudar a tomar decisiones de inversiones en mejoras de las líneas que actualmente posee un proveedor de acceso a Internet. El sistema utilizará el modelo definido para cada una de las líneas para calcular sus valores respectivos, y en base a estos valores tomar decisiones. Así, para la toma de decisiones se define el siguiente contexto:

Supóngase que se quiere analizar las inversiones para mejorar una línea de comunicaciones de un tipo determinado. Esta línea tendrá una capacidad máxima, que se denotará por \bar{Q}_1 . Además, se considera un conjunto de tipos de línea a los que la línea a estudiar se puede mejorar. Estos tipos de línea se denotarán por $2, 3, \dots, n$, ya que el tipo 1 es el tipo del que actualmente es la línea a mejorar. Supóngase también que estos n tipos de línea están ordenados de menor a mayor capacidad de transmisión. Entonces, se considera un conjunto de ecuaciones en derivadas parciales del tipo de la ecuación 2.1, uno para cada uno de los tipos de línea posibles, por lo que habrá n ecuaciones inicialmente:

$$\frac{\partial V_i}{\partial t} + \frac{1}{2}\sigma^2 Q^2 \frac{\partial^2 V_i}{\partial Q^2} + (\mu - k\sigma)Q \frac{\partial V_i}{\partial Q} - rV_i + R_i = 0, \quad \text{con } i = 1, 2, \dots, n$$

Con este conjunto de ecuaciones en derivadas parciales se puede calcular el valor de cada uno de los tipos de línea en el intervalo de tiempo a considerar. Las decisiones que se tomen se basarán en operaciones realizadas sobre los valores de los diferentes tipos de línea a considerar calculados con estas ecuaciones.

Tiempos relevantes del problema

Para encuadrar las diferentes tomas de decisión, se definen dos conjuntos de instantes temporales diferentes. Uno de los conjuntos será el de *tiempos de notificación* y el otro el de *tiempos de mejora*. Un tiempo de notificación t_n es un instante de tiempo en el cual se ha de analizar la situación para tomar una decisión sobre si mejorar la línea o no; estos tiempos formarán un conjunto de valores discretos, representando las fechas en las que la empresa propietaria de las líneas revisa el estado de su infraestructura y decide las acciones para maximizar los beneficios. Un tiempo de mejora t_u es un instante de tiempo en el que una mejora, que se había decidido en un tiempo de notificación anterior t_n , se lleva a cabo de forma efectiva. Es decir, hay un período de tiempo entre que se toma la decisión de mejorar una línea (instante t_n) y el momento en el que esa mejora resulta efectiva (instante t_u). Esta demora modela el tiempo que el proveedor del hardware tarda en entregar el nuevo equipamiento, el tiempo que se tarda en instalar y configurar, y el tiempo que cuesta formar al personal involucrado en el uso de las nuevas instalaciones (ver Figura 2.1). El modelo empleado considera que el flujo de beneficios de una red no se

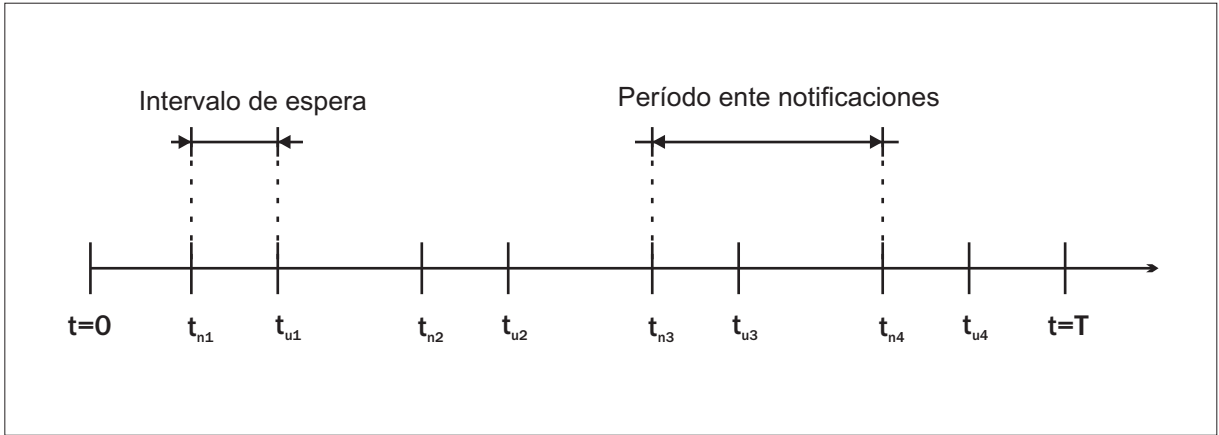


Figura 2.1: Distribución de los tiempos referentes a las decisiones

interrumpe en el período que transcurre entre el tiempo t_n y el tiempo t_u ; este comportamiento se puede cambiar de forma sencilla en el momento en el que el presente modelo no sea satisfactorio. Para establecer de forma única los tiempos del problema se define el período entre decisiones de mejora, que determina los elementos del conjunto de tiempos de notificación, y el tiempo de espera entre la decisión de mejora y la mejora efectiva, que determina el conjunto de tiempos de mejora a partir del conjunto de tiempos de notificación. Estos dos valores representan en el entorno empresarial la periodicidad de las decisiones de actualización y la duración de la implantación de las mejoras decididas, respectivamente. A estos valores se les impone la restricción de que el intervalo entre decisiones siempre será mayor o igual que el tiempo de implantación, ya que de esta forma no se solapan dos decisiones en el tiempo.

Así, la configuración de las fechas divide el dominio de la variable tiempo $([0, T])$ en una serie de subintervalos temporales, que se pueden clasificar en dos grupos:

- En primer lugar, los intervalos en los cuales la fecha más reciente es una fecha de mejora t_u y la más tardía es una fecha de notificación t_n . Estos intervalos se denominarán intervalos o zonas del tipo I. De forma general, estas zonas van a constituir las partes sencillas de la resolución del problema, donde no se realizarán cálculos referentes a las tomas de decisión.
- En segundo lugar, los intervalos en los cuales la fecha de inicio es una fecha de notificación t_n y la fecha de fin es una fecha de mejora t_u . Estas zonas del problema se definen como zonas del tipo II, y son aquéllas en las que se van a realizar los cálculos referentes a las tomas de decisión (y por ello resultarán las más costosas computacionalmente).

La distribución de estas fechas se puede observar en la Figura 2.2; en ella se observa que los dos tipos de zonas se van alternando a lo largo del tiempo. En el caso de que el tiempo entre decisiones consecutivas sea igual al tiempo de espera entre notificación y mejora, los dos conjuntos de fechas se solaparán, y una

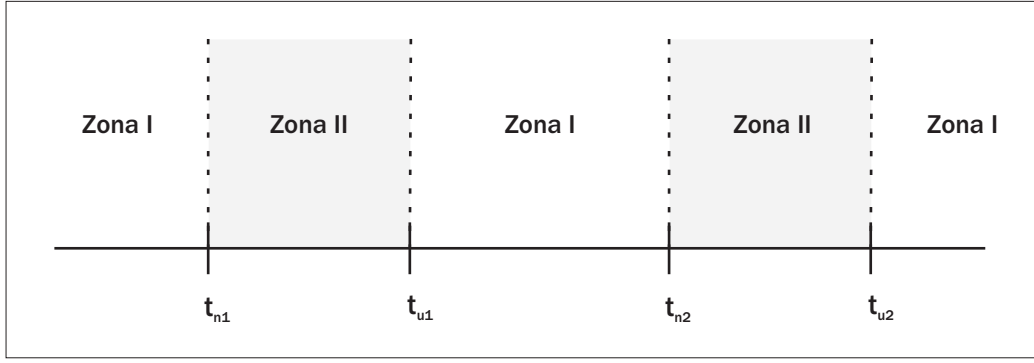


Figura 2.2: División del tiempo en subintervalos

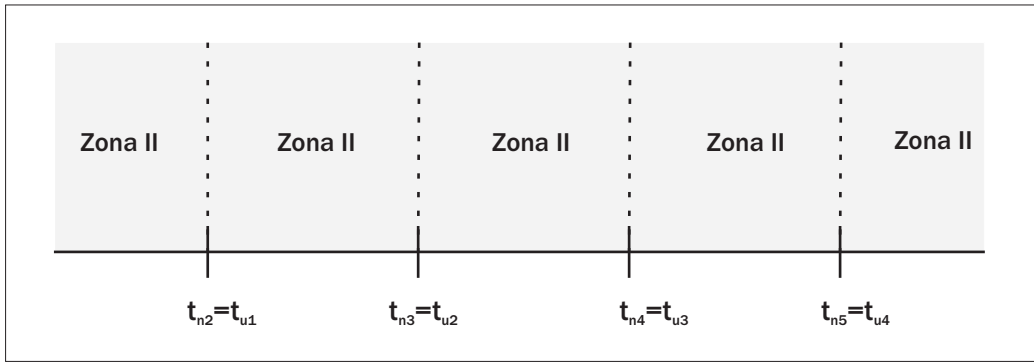


Figura 2.3: Superposición de las zonas de tipo II

fecha de mejora será igual a la siguiente fecha de decisión. En este caso, y sólo en él, el problema se compondrá exclusivamente de zonas de tipo II (ya que la longitud de las zonas de tipo I se ve reducida hasta cero). Esta situación se muestra en la Figura 2.3.

Estos dos conjuntos de fechas (notificación y mejora) son independientes del conjunto que forman las fechas de pagos de mantenimiento. Normalmente el intervalo entre pagos de mantenimiento será mucho más pequeño que el intervalo entre fechas de notificación y el tiempo de espera entre notificación y mejora. Así, será la situación habitual el que en cada una de las zonas antes definidas existan varios tiempos de pago de mantenimiento (t_{mant}), como se muestra en la Figura 2.4. Esta figura contiene un diagrama de una distribución ilustrativa de todos los tiempos relevantes del problema. En un caso real los tiempos entre mantenimientos suelen ser más pequeños que los mostrados en la figura, y están equiespaciados; además, suele haber un número mucho mayor de decisiones.

Sin embargo, los pagos de mantenimiento no afectan al algoritmo de decisión de mejoras. En lo sucesivo, las alusiones al cálculo del valor de una línea siempre se referirán a la resolución de la ecuación en derivadas parciales asociada a esa línea, *incluyendo los pagos de mantenimiento*; esto es, resolviendo la ecuación en derivadas parciales entre las fechas de pago de mantenimientos y actualizando el valor de

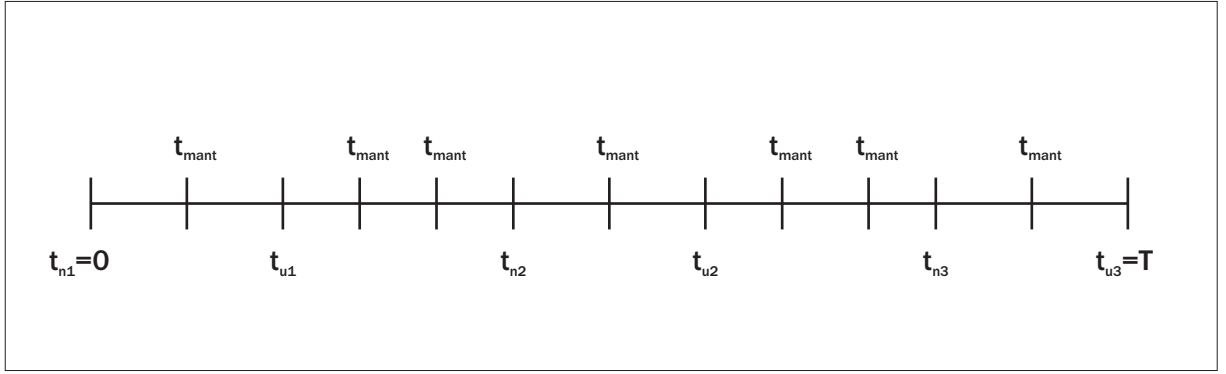


Figura 2.4: Distribución de los tiempos relevantes del problema

la línea en dichas fechas mediante la ecuación (2.5).

Esquema de resolución del problema

Una vez delimitadas las diferentes zonas del problema, se puede plantear el método de resolución del problema. El problema a resolver consiste en hallar la decisión óptima que ha de tomarse en cada uno de las fechas de notificación. La decisión puede ser o bien no mejorar la línea, o bien mejorar la línea a una de mayor capacidad (especificando el nuevo tipo). Como ya se ha mencionado, estas decisiones se tomarán en base a los valores de los diferentes tipos de líneas calculados mediante la resolución numérica del sistema de ecuaciones en derivadas parciales planteado.

El valor de cada línea se conoce en el horizonte de inversión T (recuérdese que este valor lo determina la función f). Lo que se desea conocer es el valor de esa línea en el tiempo actual ($t = 0$), de forma que la resolución de la ecuación se realizará hacia atrás en el tiempo. Esto es algo muy común en el ámbito de las opciones financieras, para las cuales se conoce el valor en el tiempo de vencimiento (que se corresponde con el horizonte de inversión en el modelo aquí propuesto) y se quiere calcular su valor en el momento actual. Así, el procedimiento que se sigue es calcular el valor de una línea en un instante t_1 a partir su valor en un instante de tiempo t_2 posterior a t_1 .

Como ya se ha mencionado antes, entre las fechas relevantes para el algoritmo de decisión (tiempos de notificación y de mejora) el valor de una línea se calcula combinando la ecuación en derivadas parciales 2.1 con la ecuación 2.5. Efectivamente, para calcular V en un tiempo inicial t_1 partiendo de un tiempo final t_2 , se pueden dar dos casos: que existan una o varias fechas de mantenimiento t_{mant} entre t_1 y t_2 o que no exista ninguna. Si no existe ninguna, para calcular $V(Q, t_1)$ (el valor V en todo el dominio de Q en el instante t_1) basta con resolver hacia atrás la ecuación en diferencias parciales 2.1 partiendo de t_2 hasta t_1 . En cambio, si hay una o más fechas de mantenimiento intermedias, el cálculo se ha de hacer desde t_2 hasta la última fecha de mantenimiento mediante la resolución de la ecuación 2.1; una vez calculado el valor *inmediatamente después* de la fecha de mantenimiento, se calcula el valor *inmediatamente antes*

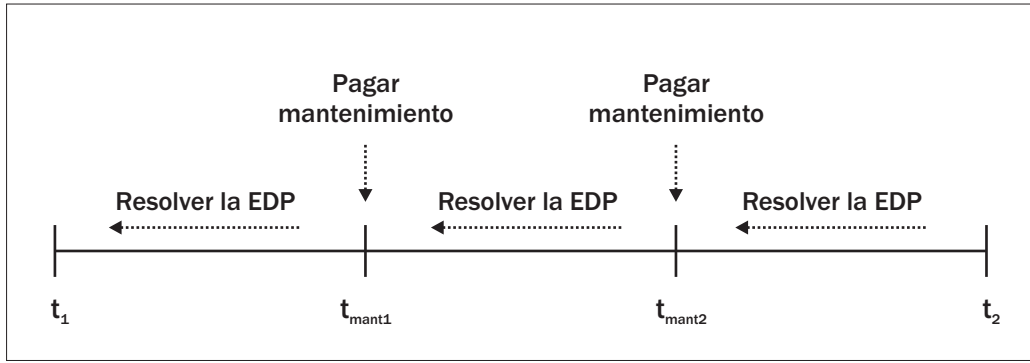


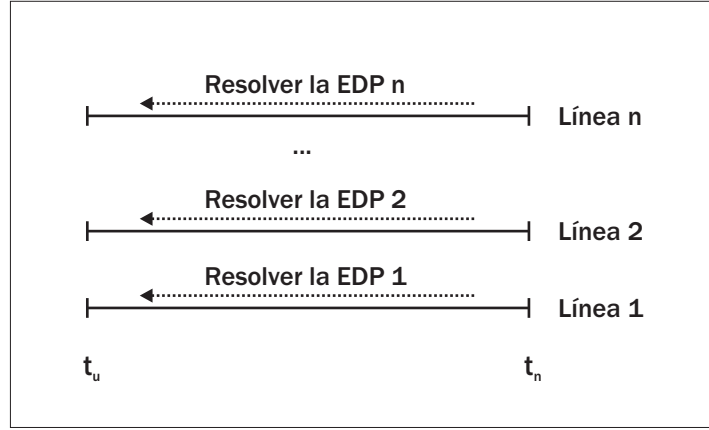
Figura 2.5: Cálculo del valor de una línea entre t_1 y t_2 considerando el mantenimiento de la misma

de dicha fecha utilizando la ecuación 2.5. De nuevo se resolverá la ecuación de la línea hasta la siguiente fecha de mantenimiento, y se actualizará el valor mediante la ecuación del pago de mantenimiento. Este procedimiento se repetirá hasta alcanzar el tiempo t_1 , donde se conocerá el valor de la línea. La Figura 2.5 ilustra el proceso.

Mediante el procedimiento anterior se resuelve el problema de calcular el valor de una línea entre dos instantes de tiempo cualquiera, considerando el pago de mantenimiento de la misma. Este es el procedimiento que se usa en el algoritmo de decisión de mejoras, de diferente forma dependiendo de si se trata de una zona de tipo I o una zona de tipo II:

- Zonas de tipo I (**problema "simple"**): en estas zonas el cálculo resulta sencillo, ya que son zonas en las cuales no se tiene que hacer ninguna consideración en cuanto a toma de decisiones se refiere. En efecto, en el tiempo entre la realización efectiva de una mejora y la toma de la siguiente decisión no ha de hacerse más que calcular el valor del conjunto de las n líneas en ese intervalo (recuérdese que el cálculo se realiza hacia atrás en el tiempo, comenzando en la fecha de notificación y terminando en la fecha de mejora). Para obtener el valor $V_i(Q, t_n)$ (el valor de la línea i en el momento de mejora, que es el extremo izquierdo de la zona), solamente hay que tomar la ecuación en derivadas parciales asociada a la línea i y resolverla numéricamente (Figura 2.5). Así, las zonas de tipo I conforman la parte menos costosa computacionalmente, ya que solamente requieren la resolución de n ecuaciones en derivadas parciales (Figura 2.6).
- Zonas de tipo II (**algoritmo de decisión**): es en estas zonas comprendidas entre una fecha de notificación y una fecha de mejora donde se emplea el algoritmo de decisión para comparar los valores de las diferentes líneas y tomar una decisión. El algoritmo se aplica a cada línea por separado, y consta de dos partes, el cálculo de los valores de mejora y la decisión entre esos valores. Suponiendo un problema con n líneas, el algoritmo de decisión para la línea i es el siguiente:

1. Cálculo de los valores de mejora: en este primer paso se calcula el valor de la línea i en la fecha

Figura 2.6: Cálculo del valor de las n líneas en una zona de tipo I

de notificación para cada una de las posibles mejoras de esa línea. Así, se calculará un valor $V_i^j(Q, t_n)$ (valor de la línea i mejorada a la línea j en el tiempo t_n) para $j = i, i+1, i+2, \dots, n$, donde el valor $V_i^i(Q, t_n)$ representa el valor de la línea i sin ser mejorada a otro tipo de línea. Para ello se toma como valor final $V_i^j(Q, t_u)$ el valor de la línea a la que se mejora $V_j(Q, t_u)$. Este es un valor conocido, y se calculó la zona anterior (bien fuese una zona de tipo I o de tipo II). Este valor final se resuelve utilizando *la ecuación de la línea i* hasta la fecha de notificación. El motivo de tomar el valor de una línea como valor final y resolverla con la ecuación de otra línea se ve claramente analizando la situación que se está modelando: en el tiempo final t_u la línea i pasa a ser una línea j si se tomó esa decisión en el tiempo de notificación. Por tanto, el valor de esa línea en ese instante del tiempo será el valor de una línea j (ya que ese valor se calcula hacia atrás en el tiempo, y por tanto no se ve afectado por la historia *pasada* de la línea, sino sólo por su historia *futura*¹). Sin embargo, en instantes anteriores a la fecha de mejora la línea todavía no es del tipo j , sino que es una línea del tipo i esperando a que sea mejorada efectivamente a una línea j , por lo que la ecuación que se use debe ser la asociada a la línea i .

2. Toma de decisiones: como entrada a este paso se tienen los valores calculados en el paso anterior. En concreto, para la línea i se habrán calculado los valores $V_i^j(Q, t_n)$ con $j = i, i+1, i+2, \dots, n$. La toma de decisión se realiza para cada uno de los puntos de Q , y consiste en elegir el valor mayor de entre todos los valores de entrada para ese q , descontando en cada caso el coste de mejorar la línea i a la línea j . Formalmente, para todo $q \in Q$,

$$V_i(q, t_n) = \max_{w=i, i+1, \dots, n} \{V_i^w(q, t_n) - C_i^j(t)\}$$

¹De esta forma, una línea de tipo i mejorada a una línea j en un instante t será indistinguible de una línea j en ese mismo instante t

Por supuesto, el coste de mejorar la línea i a sí misma es cero ($C_i^i(t) = 0$). El coste de mejora se considera dependiente del tiempo, ya que los costes del hardware van disminuyendo; este efecto se modela aplicando un factor de atenuación al coste inicial:

$$C_i^j(t) = C_i^j(0) e^{-\alpha t}$$

donde $C_i^j(0)$ es el coste *actual* de mejorar la línea i a la j . Obsérvese que el factor de atenuación utilizado es el mismo que se aplica en la función de beneficio (2.4).

La decisión dependerá por tanto de la demanda de capacidad Q que se considere; es por esto que las decisiones tendrán la forma

mejorar la línea i a la línea j si el porcentaje de utilización de la línea i supera el x % de su capacidad máxima

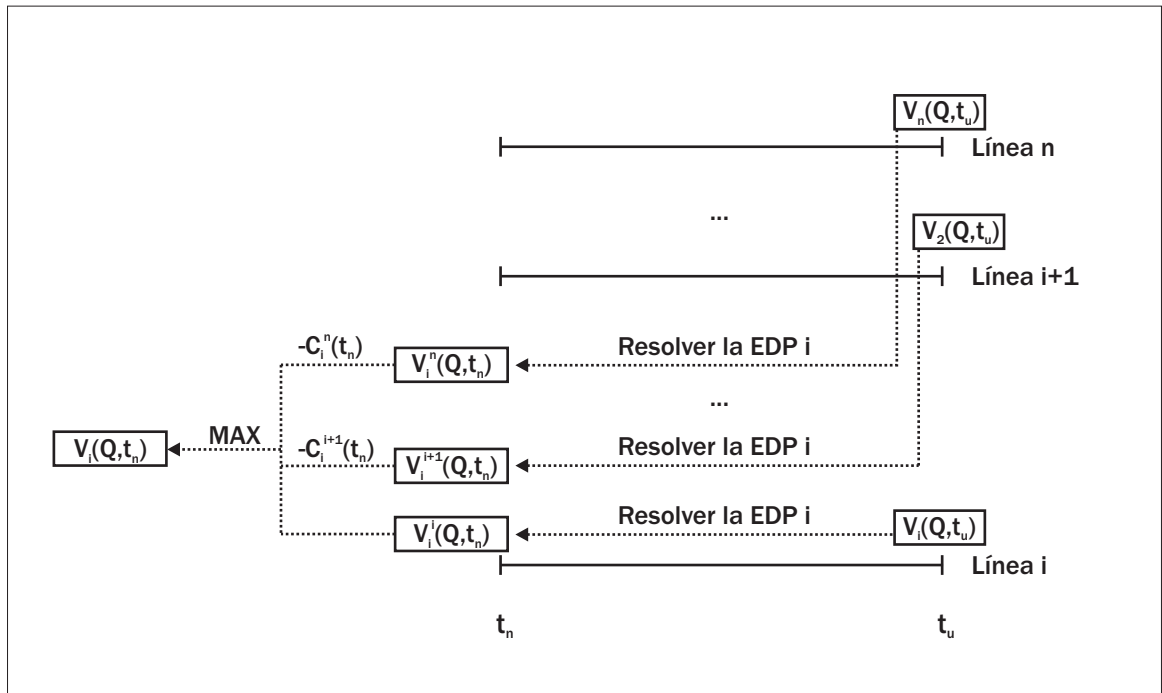
donde x corresponde al porcentaje que representa para la línea i el menor Q para el que se toma la decisión de mejorar a la línea j . De esta forma, el dominio de Q se dividirá en zonas, dependiendo de la línea a la que se mejore en cada una de ellas.

Como salida del paso 2 del algoritmo se tiene el valor $V_i(Q, t_n)$, que es el valor de la línea i en ese instante, junto con la decisión a tomar para cada uno de los posibles q . Estas decisiones se almacenarán para poder revisar las posibles decisiones a tomar en cada una de las fechas de notificación existentes. El esquema del algoritmo de decisión se muestra en la Figura 2.7.

Aplicando este algoritmo a cada una de las líneas se obtendrá el valor de cada una de ellas en la fecha de notificación y un conjunto de decisiones para cada una de las líneas en esa fecha de notificación.

Este algoritmo es mucho más costoso computacionalmente que el algoritmo aplicado en las zonas de tipo I, en el cual había que resolver n ecuaciones en derivadas parciales. En las zonas de tipo II, al contrario, habrá que resolver $n-i+1$ ecuaciones para la línea i , haciendo un total de $\frac{n(n+1)}{2} = \frac{n^2+n}{2}$ ecuaciones a resolver. Así, el algoritmo en las zonas de tipo II tiene una complejidad cuadrática en el número de líneas, frente a la complejidad lineal del algoritmo de las zonas de tipo I.

Aplicando este par de algoritmos a las zonas correspondientes del problema se irá conociendo el valor de todas las líneas en las fechas relevantes para el algoritmo de decisión (fechas de notificación y de mejora), comenzando en las fechas posteriores y terminando en $t = 0$. En este punto se obtendrá el valor de la línea 0 en la fecha actual y una tabla con las decisiones de mejora que se han de tomar para cada una de las posibilidades de utilización de la línea en cada uno de los tiempos de notificación. Por lo tanto,

Figura 2.7: Esquema del algoritmo de decisión de mejora para la línea i

se ha resuelto el problema satisfactoriamente y el propietario de la línea dispone de una guía para realizar sus inversiones en nuevas infraestructuras de la manera más favorable de acuerdo con la demanda.

Para la resolución de la ecuación en derivadas parciales se hace necesaria la utilización de métodos numéricos para ecuaciones parabólicas. En capítulos posteriores se describe la técnica de elementos finitos que se ha elegido como opción para la resolución numérica.

Capítulo 3

Modelo matemático para redes de telefonía móvil

En el presente capítulo se desarrolla el modelo empleado en la resolución del problema de dimensionamiento de redes de telefonía móvil. En principio se introduce el contexto de las operadoras de telefonía móvil, y seguidamente se introduce el modelo matemático. El modelo básico ha sido extraído de [5] y se ha ampliado para rellenar las áreas incompletas.

3.1. Introducción

En el ámbito de las redes de telefonía móvil, los operadores ofrecen sus servicios a través de una infraestructura de hardware constituida por clústers. Cada clúster es una agrupación de estaciones base, cada una de las cuales proporciona cobertura a una determinada zona, dividida en sectores. Cada una de las estaciones base ofrece el servicio a través de portadores, que pueden atender a un número de usuarios simultáneos cada uno. El aumento de la capacidad de una estación base se realiza añadiendo nuevos portadores, lo que hace que se pueda atender un mayor número de llamadas simultáneas. Sin embargo, si el clúster utiliza la tecnología CDMA (*Code Division Multiplexing Access*), una llamada puede distribuirse a lo largo de varias estaciones base simultáneamente, de forma que si se quiere aumentar la capacidad de un clúster se debe añadir el mismo número de portadores a cada una de las estaciones base. De esta forma, la capacidad se ha de aumentar de forma discreta, en pasos de capacidad igual a la de un portador.

3.2. Modelo matemático

Como en el caso de las redes de fibra óptica, en el caso de las redes de telefonía móvil el principal factor que se va a tener en cuenta será la demanda de capacidad. Sin embargo, para este segundo problema

dicha capacidad no estará medida en megabits, sino que vendrá especificada en minutos en la hora punta. La demanda, efectivamente, vendrá dada por la cantidad de minutos que se utilizan en total en la hora de mayor tráfico del día. Como en el caso anterior, y por los motivos ya expuestos, se considera la demanda como una variable que cumple la ecuación diferencial estocástica [5];

$$dQ = \mu Q dt + \sigma Q dz ,$$

siendo μ la tasa de crecimiento, σ la volatilidad, t el tiempo y dz el incremento del proceso de Wiener. La parte derecha de la ecuación se descompone en dos términos cada uno de los cuales tiene un significado. El cambio esperado en Q en un período de tiempo dt es el primero de los términos ($\mu Q dt$), la tasa de crecimiento multiplicada por el valor de Q en el intervalo dt . Sin embargo, este cambio esperado está afectado de un ruido especificado por el término $\sigma Q dz$. Cuanto más grande sea σ , mayor ruido habrá y por tanto será más difícil realizar predicciones precisas del valor de Q . La otra variable que se tendrá en cuenta en el modelo utilizado será el tiempo t . Como en el problema anterior, se utilizará el tiempo *físico* para resolver el problema, es decir, se empezará a resolver en tiempos mayores y se irá decreciendo el tiempo hasta llegar a $t = 0$; se utiliza esta aproximación en vez del cambio de variable a tiempo *inverso* porque es más intuitiva y hace que los algoritmos sean más legibles.

El tiempo final del problema vendrá especificado por el *horizonte de inversión* T ; más allá de él no se tiene en cuenta la existencia o no de fechas de decisión. El valor de una inversión en un clúster en el horizonte de inversión es un valor conocido, que dependerá de la situación real de la empresa y del hardware. Para permitir la flexibilidad en este sentido, se especifica dicho valor como

$$V(Q, T) = f(Q)$$

donde la función f determina el valor final de la inversión. En los ejemplos del presente trabajo se considerará que el valor en el tiempo T es 0 para todo Q , es decir, el clúster ha perdido todo su valor. Sin embargo, y como en el caso anterior, se pueden incluir costes o beneficios adicionales que se puedan obtener del hardware en dicho horizonte de inversión utilizando otra función f .

Como en el caso anterior, utilizando la técnica de cobertura dinámica [14] se puede plantear una ecuación en derivadas parciales de segundo orden que gobierna el valor de la inversión

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 Q^2 \frac{\partial^2 V}{\partial Q^2} + (\mu - k\sigma) Q \frac{\partial V}{\partial Q} - rV + R = 0 \quad (3.1)$$

donde V depende de las dos variables especificadas, Q y t . El parámetro r es la tasa de interés libre de riesgo involucrada en las transacciones de capital, μ es la tasa de crecimiento, σ la volatilidad, k es el precio del mercado de riesgo, que tiene el mismo significado que en el capítulo anterior, y R es la función de beneficio. Como se puede apreciar, la ecuación mostrada es la misma que aparece en el problema de la valoración de líneas de fibra óptica (ecuación 2.1), teniendo en cuenta que la variable Q ahora tiene

otras unidades. Así, el mismo método numérico se puede utilizar para la valoración de las inversiones en los dos tipos de hardware.

Como en el caso anterior, esta ecuación permite conocer el valor de una inversión en el instante actual partiendo de su valor en el horizonte de inversión, para todo el rango de valores de Q .

3.2.1. Función de beneficio

Como en el problema anterior, la función de beneficio determina el caudal de capital que recibe el poseedor del clúster de comunicación por los contratos con sus clientes. La expresión que se utiliza es la misma que se empleó en la valoración de redes de acceso a Internet:

$$R(Q, t) = \min(Q, \bar{Q}_j) P_0 e^{-\alpha t}$$

teniendo en cuenta que la longitud de la línea d no aparece en este caso ya que no tiene sentido en un clúster de telefonía móvil. En este caso \bar{Q}_j es la capacidad máxima de transmisión del clúster, P_0 es el precio de venta actual del minuto de conversación, y α es el factor de decaimiento de dicho precio de venta.

Posibilidades de arbitraje

En el presente problema se pueden hacer las mismas reflexiones acerca de las posibilidades de arbitraje que se realizaron en el problema de las líneas de acceso a Internet, por lo que no se detallan aquí (ver la sección 2.1.1 en la página 32).

Factor de seguridad

La función de beneficio anterior otorga la misma renta para todos los valores de Q por encima de la capacidad máxima de transmisión, sin tener en cuenta la degradación del servicio cuando la demanda excede la capacidad máxima. Para evitar esto, se incluye en la función de beneficio un *factor de seguridad* que incorpora la calidad de servicio en el modelo. Así, se supone un factor ϕ (en porcentaje de la capacidad máxima de transmisión) por encima del cual la calidad del servicio disminuye, y por tanto el beneficio obtenido se ve reducido, debido al cambio de operador de los clientes (por la insatisfacción producida con el actual proveedor) o a los descuentos ofrecidos a los clientes para evitar la marcha a otras compañías. La nueva función aplicando el factor $\phi \in [0, 1]$ es

$$R(Q, t) = \begin{cases} \min(Q, \bar{Q}_j) P_0 e^{(-\alpha t)} & \text{si } Q \leq \phi \bar{Q}_j \\ \bar{Q}_j \phi P_0 e^{(-\alpha t)} \max\left(1 - \frac{Q - \phi \bar{Q}_j}{\phi \bar{Q}_j}, 0\right) & \text{en caso contrario} \end{cases}$$

A partir del factor de seguridad, el beneficio disminuye hasta cero. Esta nueva función puede verse para diferentes valores de ϕ en la Figura 3.1, donde se muestra la función R sin la inclusión de la degradación de la calidad de servicio y la función R con calidad de servicio, para diferentes valores de ϕ .

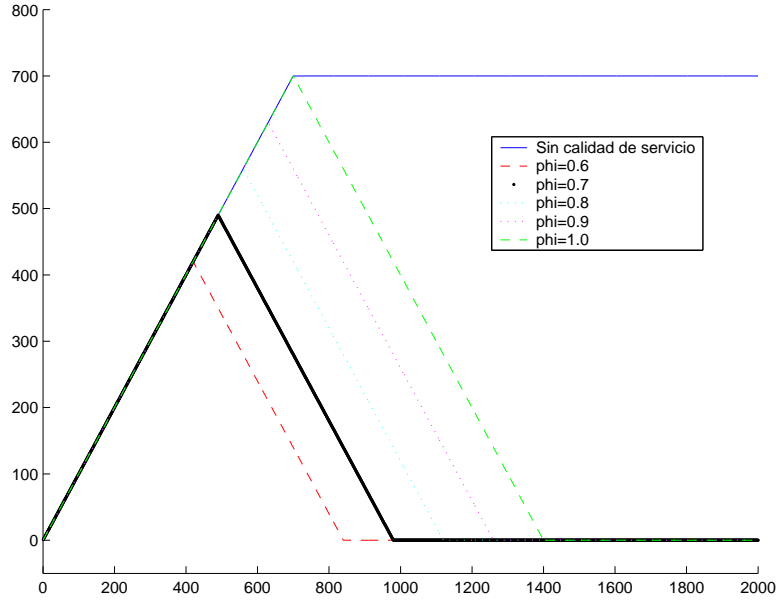


Figura 3.1: Efecto del factor ϕ en la función de renta. Valores mostrados para $\bar{Q}_j = 700$

Como en la valoración de redes de acceso a Internet, la función de beneficio R puede ser modificada para incluir nuevos factores si en algún momento no resulta satisfactoria.

3.2.2. Mantenimiento del clúster

La toma en consideración de los costes de mantenimientos de un clúster se incluye de forma muy similar al mantenimiento de las redes de fibra óptica; denotando por t^- el tiempo inmediatamente anterior a una fecha de mantenimiento y por t^+ el tiempo inmediatamente posterior a dicha fecha, se impone que

$$V(q, t^-) = V(q, t^+) - M\Delta t_m \quad (3.2)$$

siendo M el coste anual de mantenimiento y Δt_m el intervalo entre dos fechas de mantenimiento consecutivas. Se aprecia que la ecuación (3.2) es similar a la ecuación (2.5) de mantenimiento de redes de acceso a Internet.

3.2.3. Toma de decisiones

Con el marco definido hasta ahora es posible calcular el valor V de una inversión en un clúster con una capacidad determinada, partiendo de su valor en el horizonte de la inversión y resolviendo hacia atrás la ecuación en derivadas parciales (3.1), actualizando el valor mediante la ecuación (3.2) cuando se alcanzan fechas de mantenimiento, hasta llegar a $t = 0$, el instante actual.

El algoritmo de decisión está construido sobre los valores calculados para cada una de las posibles capacidades de transmisión. Así, se supone que se tiene un clúster con una capacidad de transmisión \bar{Q}_0 .

A dicho clúster se le pueden añadir portadores, de forma que se pueden alcanzar diferentes capacidades de transmisión $\bar{Q}_1, \bar{Q}_2, \dots, \bar{Q}_{j_{max}-1}$, dependiendo del número de portadores que se añada a cada estación base. Se tiene por tanto una colección de j_{max} valores máximos, que se suponen ordenados de menor a mayor. De estos valores, el primero representa la capacidad de transmisión actual del clúster y el resto representan las capacidades a las que se puede mejorar.

Tiempos relevantes del problema

A la hora de tomar las decisiones empresariales, una práctica común es la de establecer un tiempo entre revisiones de las decisiones de mejora. En el caso de las redes de telefonía móvil, este tiempo se denominará Δt_n . Definiendo este período entre revisiones, y estableciendo la primera de las revisiones en la fecha actual ($t = 0$), quedan fijadas todas las fechas de notificación t_n del problema.

Cuando el hardware se encarga, tarda un tiempo en ser servido. Además, la instalación también requiere tiempo, como la configuración para el óptimo funcionamiento. Este tiempo se denomina γ y constituye el intervalo entre que se toma la decisión de mejora (t_n) y que la mejora es efectiva (t_u). A partir de las fechas de notificación se obtienen todas las fechas de mejora del problema.

A la hora de resolver el problema se define una serie de *tiempos de observación*, que son aquellos tiempos en los que se actualizan los valores calculados como consecuencia de alcanzar una fecha relevante del problema. Así, estas fechas de observación se van a producir cada Δt_{obs} . Cada una de estas fechas podrá ser una fecha señalada del problema (fecha de notificación t_n , fecha de mejora t_u o fecha de pago de mantenimiento t_m), o podrá no ser de ninguno de esos tipos. Dado que solamente se permite que las fechas relevantes estén emplazadas en tiempos de observación, se imponen las restricciones

$$\frac{\Delta t_n}{\Delta t_{obs}}, \frac{\Delta t_m}{\Delta t_{obs}} \text{ y } \frac{\gamma}{\Delta t_{obs}} \text{ son números enteros}$$

que junto con la restricción de que todas las fechas relevantes comienzan en $t = 0$, hacen que dichas fechas relevantes coincidan en fechas de observación.

De nuevo, estas fechas dan lugar a la aparición de dos tipos de zonas, las zonas de tipo I y las de tipo II, que conservan su significado con respecto a la valoración de líneas de fibra óptica. Las zonas de tipo I serán aquellas en las que no hay ninguna mejora parcialmente completada, mientras que las zonas de tipo II serán las que contienen alguna mejora que no ha sido completada (por lo que estas zonas estarán entre un t_n y el t_u siguiente). En el presente problema se calculan las tomas de decisión de una forma más flexible que en el caso de las redes de fibra óptica. En este caso no se requiere que $\Delta t_n \geq \gamma$, que era la restricción que hacía que las zonas de tipo II nunca se superpusieran (solamente podían unirse, pero no superponerse). En este modelo se elimina esta restricción, permitiéndose que se superpongan diferentes zonas de tipo II, haciendo que en algunos momentos se tenga que llevar la cuenta de varias tomas de decisión coincidentes en el tiempo. Esta situación se refleja en la Figura 3.2, en la que en ciertos momentos

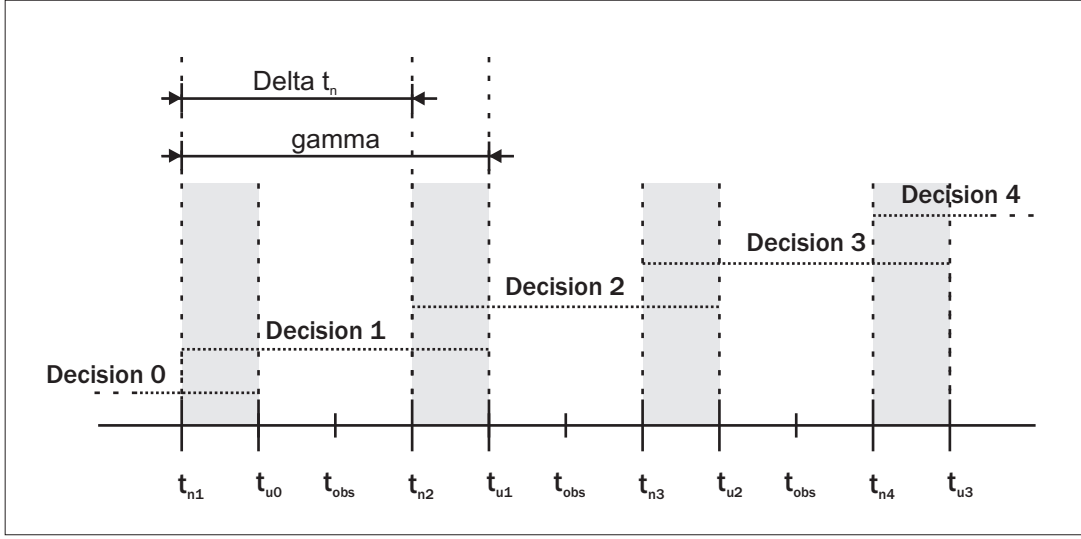


Figura 3.2: Superposición de varias zonas de tipo II cuando γ es mayor que Δt_n . Las zonas en gris corresponden a los intervalos de superposición de varias zonas de tipo II

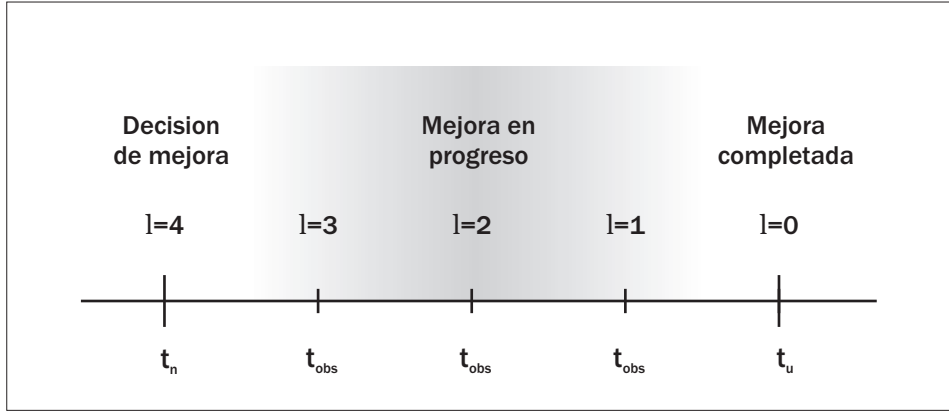
se solapan dos decisiones. Sin embargo, podrían solaparse más zonas si γ fuese lo suficientemente grande con respecto a Δt_n .

Mejoras incompletas

Dado que en este modelo en un instante de tiempo t pueden estar evaluándose varias decisiones, será necesario almacenar en ese instante el valor de los diferentes clústers, junto con el valor en t de cada una de las posibles mejoras *para cada una de las decisiones que se están evaluando en t* . Es decir, se añade un nuevo nivel de complejidad al algoritmo, al tener en cuenta el *porcentaje de mejora realizado de cada posible mejora*. Dado que solamente se van a realizar modificaciones puntuales a los valores de los clústers en las fechas de observación, se definen una serie de posibles niveles de realización para las mejoras. La cantidad de estos niveles es $l_{max} + 1$, donde l_{max} se calcula como

$$l_{max} = \frac{\gamma}{\Delta t_{obs}} \quad (3.3)$$

Esto es razonable, ya que $l_{max} + 1$ es el número de fechas de observación comprendidas entre una fecha de notificación y una fecha de mejora (si se incluyen ambas, que también son fechas de observación). Así, una mejora comenzará en $l = l_{max}$ cuando se tome la decisión de mejorar el clúster. Cuando las fechas de observación vayan transcurriendo, el valor de l irá decreciendo a medida que la mejora se completa, hasta alcanzar el valor 0 cuando $t = t_u$. En cada una de las fechas de observación t_{obs} el valor de l se actualizará de la siguiente forma

Figura 3.3: Evolución del valor de l para una mejora

$$l(t_{obs}^-) = l(t_{obs}^+) - 1 \quad (3.4)$$

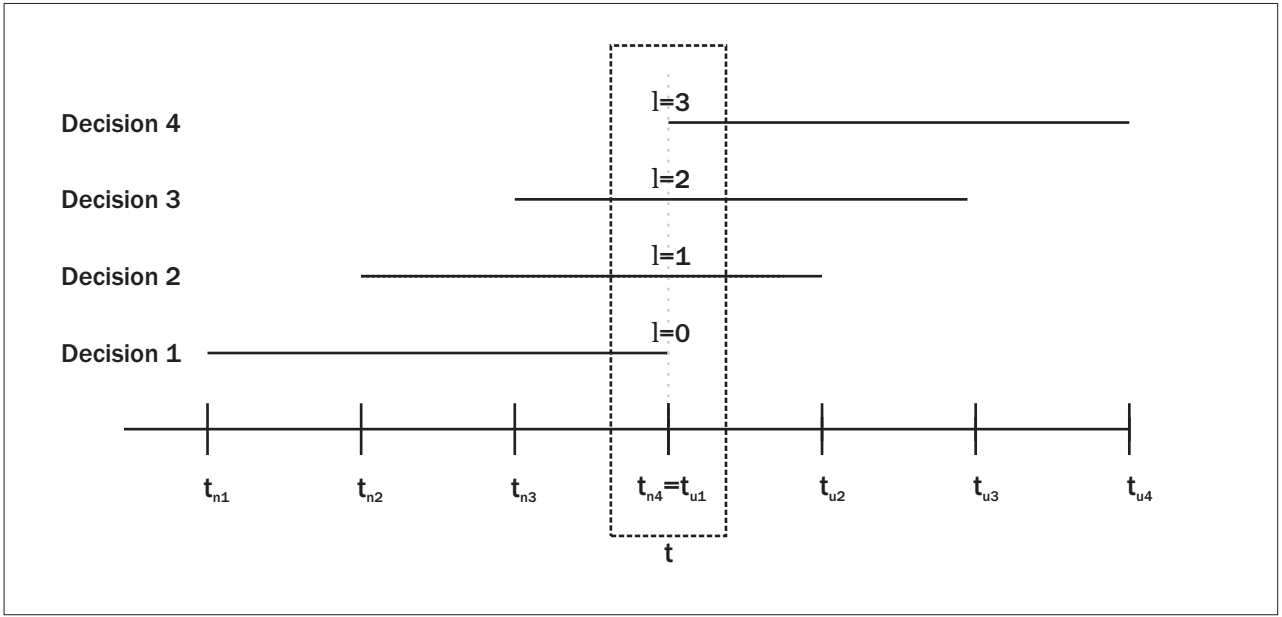
El valor de l puede entenderse como el número de fechas de observación restantes hasta que la mejora sea efectiva. Esta evolución se muestra en la Figura 3.3

El valor de la variable l en un instante t será diferente para los valores asociados a diferentes tomas de decisión que están superpuestas; las mejoras que se hayan decidido después tendrán valores mayores de l que las que se decidieron antes, y, por lo tanto, están más proximas a su realización efectiva (Figura 3.4).

La distribución de la información que existe en este problema se muestra en la Figura 3.5. En dicha figura se observa la existencia de tres ejes, uno para cada parámetro que determina los diferentes valores existentes, a saber, j (que determina el clúster inicial), u (que determina a qué clúster se mejora el clúster j) y l (que determina el grado de realización de dicha mejora). Así, solamente tendrán sentido algunas de las posibles combinaciones de los tres parámetros, ya que se tienen que cumplir algunas restricciones:

- Con $l = 0$, solamente serán válidos los valores $V_{x,x,0}$, ya que $l = 0$ se utiliza para almacenar el valor del clúster x sin mejorar.
- Para cada j se tienen valores V solamente para $u > j$, ya que los clústers están ordenados de menor a mayor capacidad, y solamente se puede mejorar un clúster a otro de mayor capacidad de transmisión.

Estas dos restricciones hacen que solamente algunas de las casillas mostradas en la figura se utilicen. Cada una de esas casillas representa un vector de valores conteniendo el valor del clúster para todo el dominio de Q . En concreto, las casillas que se utilizan son las marcadas en gris claro para los valores de los diferentes clústers sin mejorar y las marcadas en gris oscuro para los valores de los clusters j mejorados a otros clústers u .

Figura 3.4: Diferentes valores de l para diferentes mejoras superpuestas

Al haber introducido la capacidad de considerar en los tiempos del modelo las mejoras no completadas, se puede añadir la capacidad de realizar los pagos de mejora de forma fraccionada. En vez de pagar todo el coste de la mejora en el momento de la decisión, se puede dividir el pago en varios plazos, uno en cada fecha de observación entre la fecha de decisión y la fecha de mejora. Para especificar la cantidad a pagar en cada uno de los plazos se define la función C . Esta función determina el coste del plazo a pagar para una mejora de la línea j a la línea u , con un estado de realización de la mejora x a otro estado y . Por ello la notación completa utilizada será

$$C_{j \rightarrow u}(t)_{x \rightarrow y}.$$

Los plazos a pagar se realizarán en todas las fechas de observación entre la fecha de decisión t_n (incluida) y la fecha de mejora t_u (no incluida). Considerando el tiempo físico, el primer pago se hará en la primera fecha de observación antes de la fecha de mejora, $t_u - t_{obs}$ y el pago será de $C_{j \rightarrow u}(t)_{2 \rightarrow 1}$. El último de los pagos será en la fecha de notificación t_n y su cantidad vendrá determinada por $C_{j \rightarrow u}(t)_{0 \rightarrow l_{max}}$. En ese momento la función pasa del estado 0 al estado l_{max} , ya que en la fecha de decisión se pasa de una línea en estado 0 (cuya mejora está completada, que es el caso, porque la línea no está siendo mejorada a ninguna otra en el momento t_n^-) a un estado l_{max} , debido a que se toma la decisión de mejorar (en el caso de tomarla) y por tanto la línea pasa a estar en el estado l_{max} , quedándole l_{max} meses hasta ser completada. El valor $l = 0$ se utilizará para denotar el valor de una línea que no está siendo mejorada. Definiendo diferentes expresiones para C se pueden modelar diferentes formas de pago. Por ejemplo, pagar todo el coste en el momento de la notificación, pagar el coste repartido uniformemente entre los plazos,

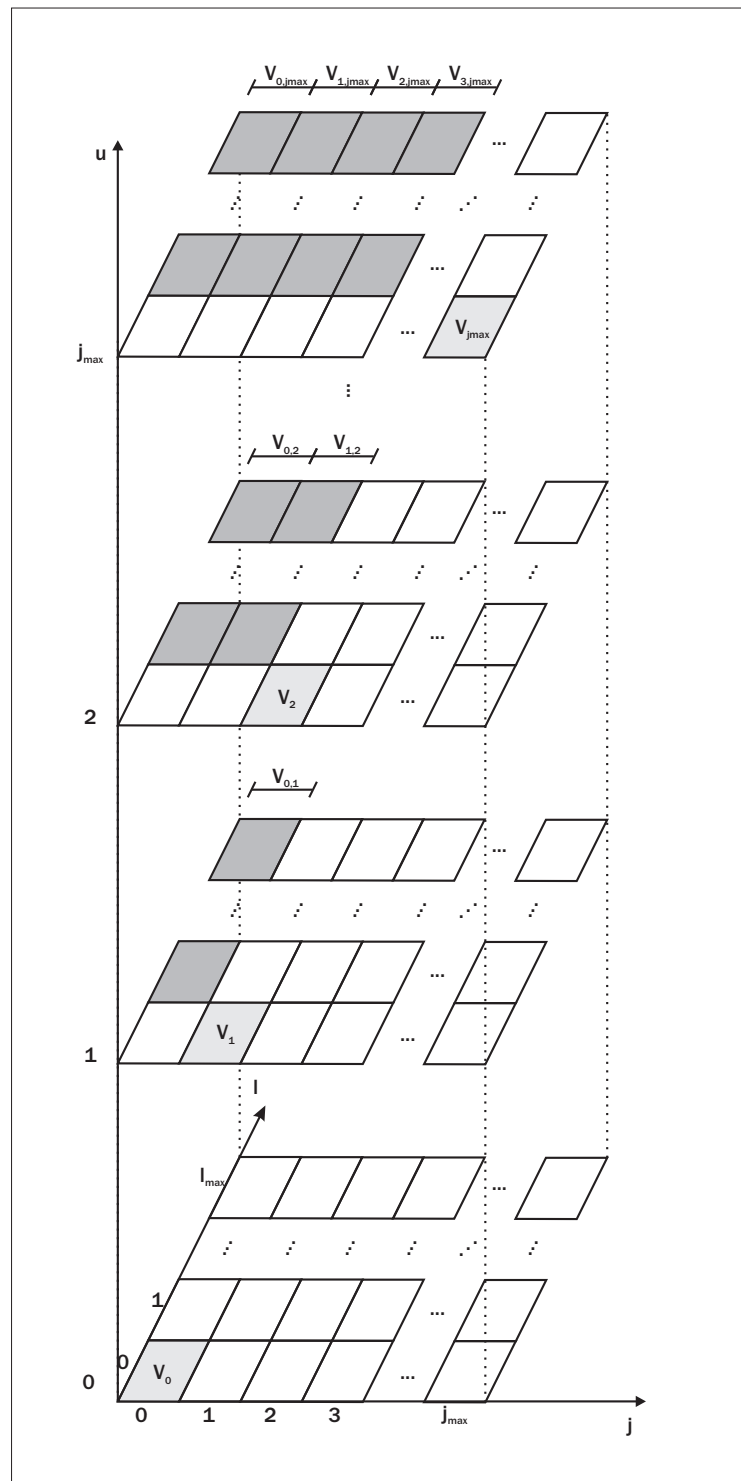


Figura 3.5: Distribución de la información en el problema

pagar el coste repartido entre los plazos de forma no uniforme, etc. Esto aporta una mayor flexibilidad a la hora de calcular las decisiones óptimas, mejorando de forma sustancial la calidad de los resultados que obtendrá el modelo.

De esta forma, en cada una de las fechas de observación en las cuales se esté calculando una toma de decisión (las comprendidas entre una fecha de notificación y una fecha de mejora) se deberá actualizar todos los valores de las líneas mediante la ecuación

$$V_{u,j,l}(Q, t_{obs}^-) = V_{u,j,l}(Q, t_{obs}^+) - C_{j \rightarrow u}(t)_{l \rightarrow l-1} \quad (3.5)$$

y en la fecha de notificación, la expresión utilizada será

$$V_{u,j,l_{max}-1}(Q, t_n^+) - C_{j \rightarrow u}(t)_{0 \rightarrow l_{max}} \quad (3.6)$$

para calcular el valor de la línea j mejorada a la línea u en la fecha de notificación, que será el valor que se empleará en el algoritmo de toma de decisión.

Toma de decisiones

Para resolver el problema de la valoración de inversiones en redes de telefonía móvil se necesita definir el algoritmo de decisión de mejora, que será similar al definido para las redes de fibra óptica. El problema está definido en el intervalo de tiempo $[0, T]$, pero se subdivide en zonas de dos tipos: de tipo I y de tipo II. En este caso, las zonas de tipo I son aquellas en las que no se está evaluando ninguna decisión de mejora, es decir, que están comprendidas entre un t_u y un t_n posterior, y no están entre ningún t_n y t_u . Las zonas de tipo II, en cambio, serán aquellas en las que está evaluando una mejora, por lo que están comprendidas entre *algún* t_n y su t_u asociado, pudiendo estar en varios intervalos de decisión, debido al posible solapamiento, como ya se mencionó anteriormente. El número de zonas de cada tipo vendrá determinado por varios factores, como son: Δt_n , que indicará la frecuencia de las tomas de decisión, y, principalmente, del cociente $\frac{\gamma}{\Delta t_n}$, que indicará si hay solapamiento de intervalos de decisión o no. En el caso de que los haya, entonces no habrá ninguna zona de tipo I, ya que cuando se llegue a una fecha de notificación t_n (resolviendo el problema hacia atrás en el tiempo) ya se habrá sobrepasado otra fecha de mejora t_u , por lo que siempre se estará evaluando una mejora (ver Figura 3.2). Así, se propone un cálculo diferente para cada una de las zonas:

- Zonas de tipo I : en estas zonas el problema se resuelve de la misma manera que se resolvía en las redes de fibra óptica. Así, para una zona comprendida entre una fecha de mejora t_u y una fecha de notificación t_n , se comenzará a resolver la ecuación en derivadas parciales asociada a cada uno de los clústers, recorriendo el tiempo en sentido inverso, hasta llegar a una fecha de observación t_{obs} . Si esta fecha es la fecha t_u , entonces se ha completado el cálculo en esa zona. Si dicho t_{obs} es una

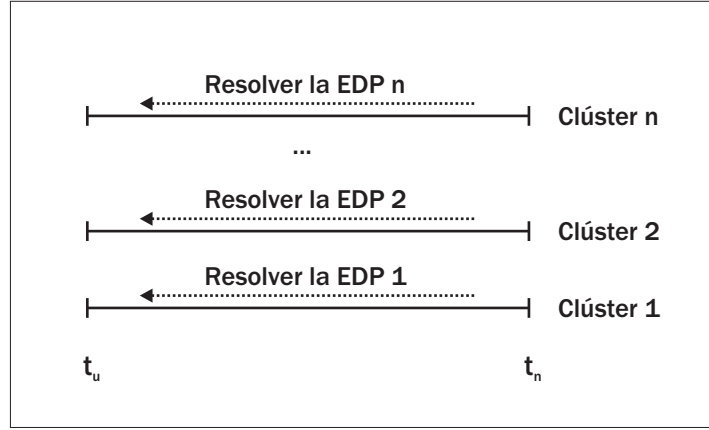


Figura 3.6: Cálculo del valor de los n clústers en una zona de tipo I

fecha de mantenimiento, entonces se habrá de actualizar el valor del clúster según la ecuación 3.2, y se seguirá resolviendo la ecuación hacia atrás en el tiempo. Así se obtendrá el valor de cada uno de los clústers en el inicio de la zona I, que es una fecha de mejora t_u (ver Figura 3.6).

- Zonas de tipo II : en estas zonas el problema se vuelve más costoso computacionalmente. Como en el caso de las redes de fibra óptica, el algoritmo no se divide en dos pasos claros como en el caso de las redes de fibra óptica (el cálculo de los valores de mejora y la toma de decisión), sino que al llevarse las tomas de decisión de varias mejoras al mismo tiempo, los dos pasos se entrelazan. La resolución en toda la zona se divide en resoluciones para cada intervalo entre fechas de observación pertenecientes a la zona. Entre dos de estas fechas la única operación que hay que realizar es la misma que en una zona de tipo I, es decir, resolver en ese intervalo $[t_{obs1}, t_{obs2}]$ todo el conjunto de ecuaciones en derivadas parciales que se almacena, para cada uno de los clústers, para cada una de las posibilidades de mejora y para cada uno de los posibles estados de dicha mejora. El valor de las diferentes líneas se tendrá que actualizar al llegar a cada uno de los diferentes t_{obs} , y las operaciones a ejecutar dependerán del tipo de fecha que sea ese t_{obs} :

- Si t_{obs} es una fecha de mantenimiento entonces habrá que realizar el pago de cada uno de los clústers, según (3.2).
- Las decisiones para las que t_{obs} no coincide ni con su t_n ni con su t_u han de ser actualizadas, ya que se está en un grado de terminación parcial de dicha mejora, por lo que han de moverse un grado más hacia su inicio (dado que se está resolviendo hacia atrás en el tiempo, y se está calculando la decisión desde el momento en el que está completa hacia el momento en el que se toma su decisión). Así, para los valores de los clústers que estén siendo mejorados a otro clúster y cuyo l sea mayor que 0 y menor que l_{max} , habrá que actualizarlos a un paso más lejos de su terminación. Para ello lo que se hará es realizar el pago parcial de la mejora

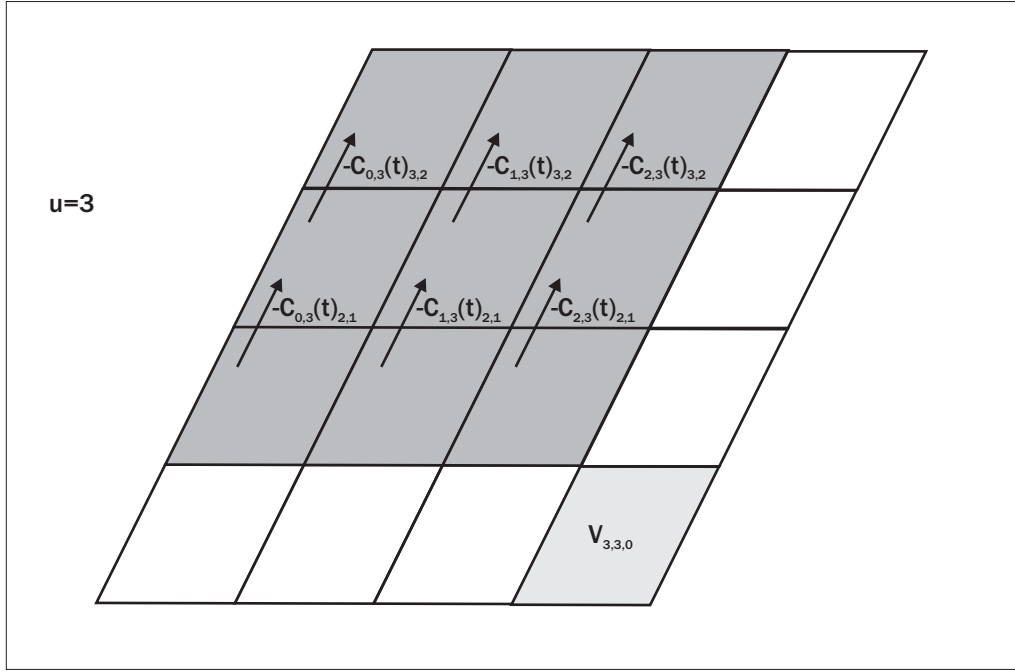


Figura 3.7: El flujo de información para decisiones parcialmente completadas

determinado por la función C de ese clúster

$$V_{j,u,l}(Q, t_{obs}^-) = V_{j,u,l-1}(Q, t_{obs}^+) - C_{j \rightarrow u}(t)_{l \rightarrow l-1}$$

para $l = 2, 3, \dots, l_{max}$. De esta manera, todas las mejoras se moverán a un paso más hacia su inicio. Sin embargo, los valores de la mejoras que en t_{obs}^+ estaban en su estado inicial $l = l_{max}$ son sobrescritos, de forma que han de ser guardados temporalmente en una serie de valores W para poder utilizarlos (estos son los valores de las decisiones para las cuales t_{obs} coincide con su t_n , y serán utilizados en las operaciones descritas más adelante). Es decir, se construyen:

$$W_{j,u}(Q, t_{obs}^-) = V_{j,u,l_{max}}(Q, t_{obs}^+)$$

Un ejemplo del flujo que sigue la información para $u = 3$, $l_{max} = 3$ se puede ver en la Figura 3.7.

- Si $t_{obs} = t_{u_w}$ quiere decir que se está en el final de una toma de decisión, de forma que en ese punto hay que comenzar a tomar en cuenta la nueva decisión. Para ello, se tiene que tomar un valor inicial para $V_{j,u,1}$, que es el valor del clúster j mejorado al clúster u en el estado en el que todavía le falta un Δt_{obs} para ser completado. Este valor se toma como el valor del clúster u en $t = t_{obs}^+$ sin mejorar, es decir, $V_{u,u,0}(Q, t_{obs}^+)$. Así, para cada una de las posibles mejoras que terminan en t_{u_w} se tomará como valor inicial

$$V_{j,u,1}(Q, t_{obs}^-) = V_{u,u,0}(Q, t_{obs}^+)$$

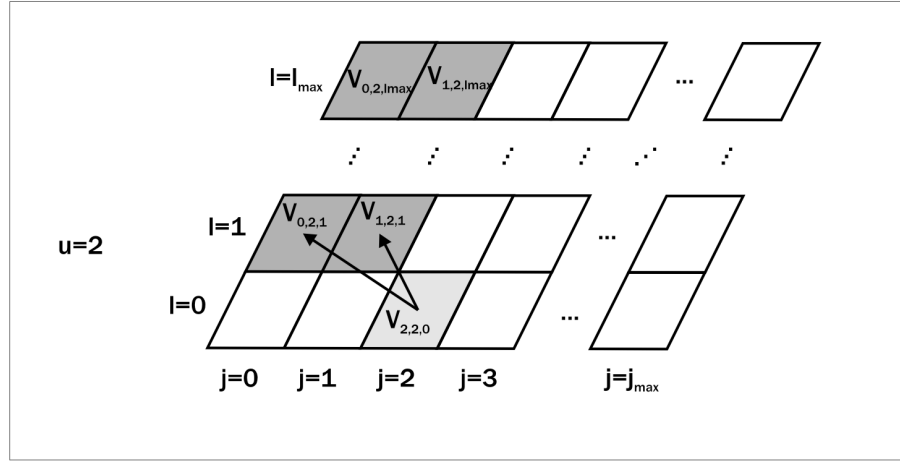


Figura 3.8: El flujo de información para nuevas decisiones (mejoras completadas)

Recuérdese que los valores de $l = 0$ se utilizan para almacenar el valor de un clúster sin mejorar a ningún otro, y que no se realiza ningún pago de mejora en el momento final de la misma. Un ejemplo del flujo de información para un valor de $u = 2$ se puede observar en la Figura 3.8

- Si $t_{obs_2} = t_{n_w}$ quiere decir que se ha alcanzado el principio de alguna toma de decisión de entre las incluidas en la zona II (recuérdese que en la zona se pueden solapar varias decisiones). Para tomar la decisión en ese punto se compara el valor del clúster j con los valores del clúster j mejorado a cada uno de los posibles clústers u que hayan sido completados. Estos valores son los W almacenados anteriormente, y con ellos se actualiza el valor del clúster j :

$$V_{j,j,0}(q, t_{n_w}^-) = \max\{V_{j,j,0}(q, t_{n_w}^+), \max_{u=j+1, \dots, j_{max}} \{W_{j,u}(q, t_{n_w}) - C_{j \rightarrow u}(t_{n_w})_{0 \rightarrow l_{max}}\}\}$$

Al valor temporal del clúster j mejorado al clúster u ($W_{j,u}$) hay que restarle el primero de los plazos de mejora que se van a realizar (y que corresponde al pago de pasar del estado 0 al estado l_{max}). El valor resultante en cada q será el máximo de todos los valores de entre el valor del clúster sin mejorar ($V_{j,j,0}$) y los valores del clúster mejorado actualizados (W); cuando el valor elegido sea el de una de las posibles mejoras entonces se almacenará la decisión tomada, que tendrá la forma

mejorar el clúster j al clúster u si el porcentaje de utilización del clúster j supera el $x\%$ de su capacidad máxima

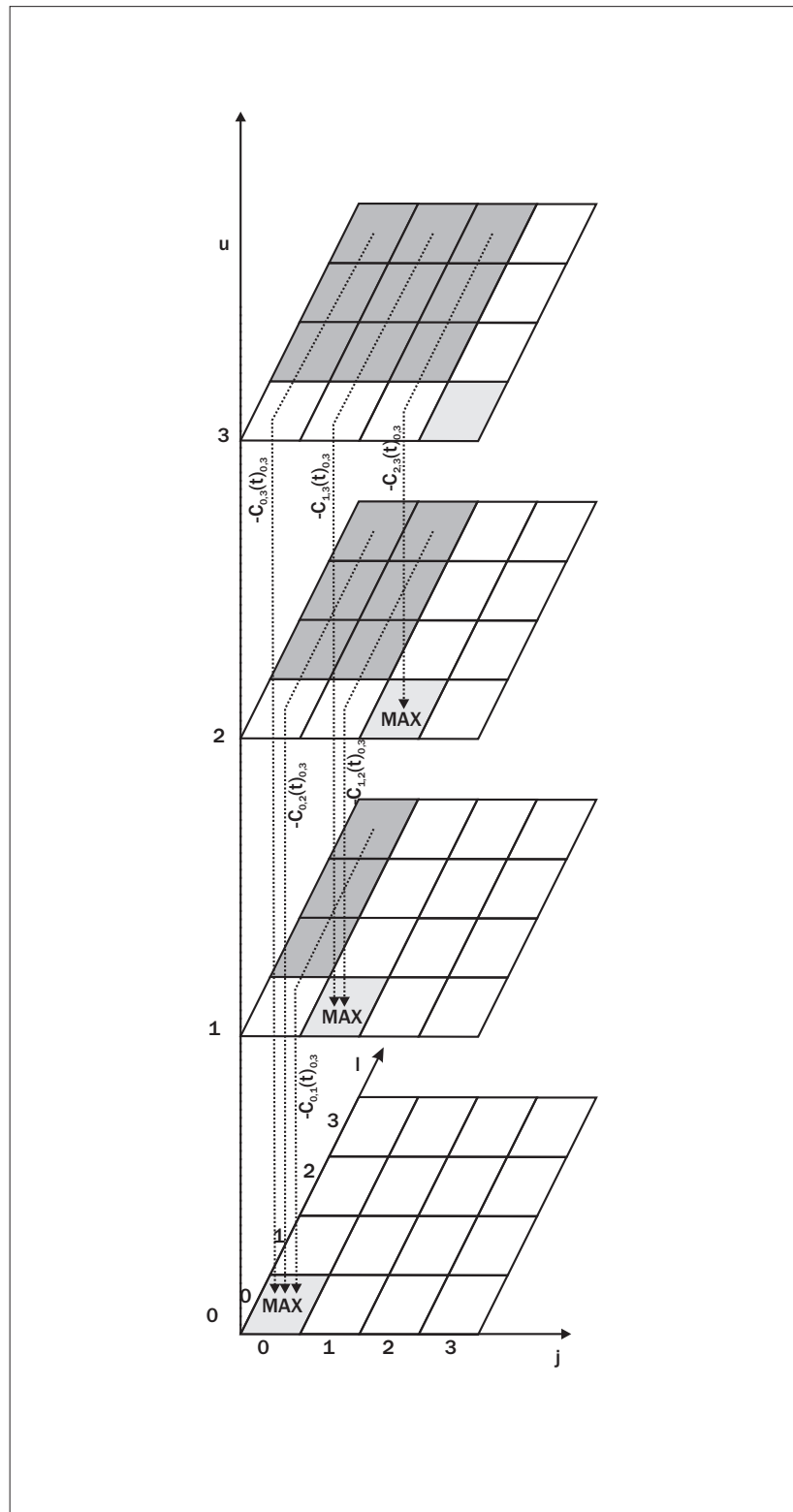
con x el porcentaje que representa para el clúster j el menor q para el que se toma la decisión de mejorar al clúster u . Un ejemplo del flujo de información para las decisiones con cuatro

clústers y $l_{max} = 3$ se muestra en la Figura 3.9

Generalmente no se utilizará el algoritmo del cálculo de las zonas de tipo I, ya que el tener la posibilidad de solapar varias decisiones será normal el que se utilice esta característica; dado que si se solapan dos decisiones no habrá ninguna zona de tipo I, normalmente no se utilizará el cálculo de estas zonas.

De esta forma se irán almacenando todas las decisiones que el sistema alcance, y se podrá conocer en cualquier fecha de notificación la acción a tomar para cada posible Q ; el cálculo se comenzará en $t = T$ y se realizará hacia atrás, calculando cada vez fechas más recientes, hasta $t = 0$.

En el capítulo siguiente se describirá el método de elementos finitos utilizado para resolver las ecuaciones en derivadas parciales entre las fechas de observación, necesario dada la naturaleza de las ecuaciones.

Figura 3.9: El flujo de información para decisiones en t_n

Capítulo 4

Derivación del método de elementos finitos

En este capítulo se presenta el desarrollo efectuado, a partir de la ecuación en derivadas parciales planteada en las secciones anteriores, para obtener el método numérico que resuelve la ecuación en derivadas parciales. El uso de métodos numéricos es obligado en la resolución de la ecuación que surge del modelo matemático del problema de valoración de inversiones en redes, ya que para esta ecuación no se puede calcular una solución exacta de forma analítica. Así, se deben emplear métodos numéricos para aproximar la solución de la ecuación. El método numérico presentado es válido para la resolución de los dos problemas objeto del presente Proyecto Fin de Carrera, ya que ambos problemas dan lugar a la misma ecuación parabólica. .

En el caso de la valoración de inversiones en redes, la ecuación resultante depende de dos variables, el ancho de banda (Q)¹ y el tiempo (t). Como se trata de una ecuación parabólica para una función dependiente de esas variables, primero se discretizará la misma en el tiempo mediante el método de características y a continuación se discretizará en la variable capacidad mediante elementos finitos de Lagrange de grado 1.

En primer lugar, se reescala la variable capacidad para que esté normalizada en el intervalo $[0, 1]$.

4.1. Reescalado de la ecuación

Partiendo de la ecuación modelo que surge en ambos problemas:

¹como ya se ha explicado previamente, el ancho de banda en las redes de un proveedor de acceso a internet está especificado en Megabits, mientras que el ancho de banda en redes de telefonía móvil está especificado en minutos de tráfico en la hora punta

$$\frac{\partial V}{\partial t} + (\mu - \sigma k)Q \frac{\partial V}{\partial Q} + \frac{1}{2}\sigma^2 Q^2 \frac{\partial^2 V}{\partial Q^2} - rV + R = 0 \quad (4.1)$$

Para que la malla de elementos finitos no dependa de la capacidad máxima, se propone el siguiente cambio de variable (normalización):

$$\tilde{Q} = \frac{Q}{Q_{max}} \quad \Rightarrow \quad Q = \tilde{Q}Q_{max}$$

Este cambio de variable hace que el nuevo dominio en ancho de banda normalizado (\tilde{Q}) sea $[0, 1]$ en lugar de $[0, Q_{max}]$, lo que permite que especificando un paso de ancho de banda (h) se pueda aplicar con el mismo significado a líneas de diferente ancho de banda máximo.

Con la nueva variable, las derivadas que intervienen en (4.1) se reescriben en la nueva variable \tilde{Q} en la forma:

$$\frac{\partial V}{\partial Q} = \frac{\partial V}{\partial \tilde{Q}} \frac{\partial \tilde{Q}}{\partial Q} = \frac{1}{Q_{max}} \frac{\partial V}{\partial \tilde{Q}} \quad (4.2)$$

$$\frac{\partial^2 V}{\partial Q^2} = \frac{\partial}{\partial Q} \left(\frac{\partial V}{\partial Q} \right) = \frac{\partial}{\partial \tilde{Q}} \left(\frac{1}{Q_{max}} \frac{\partial V}{\partial \tilde{Q}} \right) \frac{\partial \tilde{Q}}{\partial Q} = \frac{\partial}{\partial \tilde{Q}} \left(\frac{1}{Q_{max}} \frac{\partial V}{\partial \tilde{Q}} \right) \frac{1}{Q_{max}} = \frac{1}{Q_{max}^2} \frac{\partial^2 V}{\partial \tilde{Q}^2} \quad (4.3)$$

y la nueva función \bar{R} , definida en términos de \tilde{Q} en lugar de Q toma la expresión

$$\bar{R}(\tilde{Q}, t) = \begin{cases} Q_{max} \tilde{Q} d P_0 e^{-\alpha t} & \text{si } Q \leq \bar{Q}_i = \tilde{Q} \leq \frac{\bar{Q}_i}{Q_{max}} \\ 0 & \text{en caso contrario} \end{cases}$$

Sustituyendo estos nuevos valores en (4.1) se obtiene:

$$\frac{\partial V}{\partial t} + (\mu - \sigma k) \tilde{Q}Q_{max} \frac{\partial V}{\partial \tilde{Q}} \frac{1}{Q_{max}} + \frac{1}{2}\sigma^2 \tilde{Q}^2 Q_{max}^2 \frac{\partial^2 V}{\partial \tilde{Q}^2} \frac{1}{Q_{max}^2} - rV + \bar{R} = 0$$

o, equivalentemente:

$$\frac{\partial V}{\partial t} + (\mu - \sigma k) \tilde{Q} \frac{\partial V}{\partial \tilde{Q}} + \frac{1}{2}\sigma^2 \tilde{Q}^2 \frac{\partial^2 V}{\partial \tilde{Q}^2} - rV + \bar{R} = 0 \quad (4.4)$$

que es el mismo tipo de ecuación que (4.1), con la salvedad de que la expresión de la función \bar{R} es diferente.

4.2. Escritura en forma conservativa

A continuación se ha de modificar la ecuación a su expresión como ley de conservación para adaptarla al método de elementos finitos con características. Utilizando la fórmula para la derivada del producto se tiene

$$\frac{\partial}{\partial \tilde{Q}} \left(\frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial V}{\partial \tilde{Q}} \right) = \sigma^2 \tilde{Q} \frac{\partial V}{\partial \tilde{Q}} + \frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial^2 V}{\partial \tilde{Q}^2},$$

o, equivalentemente,

$$\frac{\partial}{\partial \tilde{Q}} \left(\frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial V}{\partial \tilde{Q}} \right) - \sigma^2 \tilde{Q} \frac{\partial V}{\partial \tilde{Q}} = \frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial^2 V}{\partial \tilde{Q}^2} \quad (4.5)$$

Teniendo en cuenta que la parte derecha de (4.5) aparece en (4.4), se obtiene

$$\frac{\partial V}{\partial t} + (\mu - \sigma k - \sigma^2) \tilde{Q} \frac{\partial V}{\partial \tilde{Q}} + \frac{\partial}{\partial \tilde{Q}} \left(\frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial V}{\partial \tilde{Q}} \right) - rV + \bar{R} = 0, \quad (4.6)$$

que es una forma mas apropiada para la aplicación del método de características para discretizar en tiempo y elementos finitos en la variable demanda.

4.3. Condiciones de contorno

Las condiciones de contorno son las siguientes:

- Para $\tilde{Q} = 0$:

Para la obtención de la condición de contorno que se utilizará en $\tilde{Q} = 0$ basta con sustituir en (4.6) el valor $\tilde{Q} = 0$, y desarrollar el resultado. En concreto, se obtiene

$$\frac{\partial V}{\partial t} - rV = 0, \text{ en } \tilde{Q} = 0 \quad (4.7)$$

ya que $\bar{R}(0, t) = 0$.

- Para $\tilde{Q} = 1$:

En el extremo derecho del dominio de \tilde{Q} no se puede encontrar de forma analítica una condición de contorno aplicable, pero de forma experimental se ha comprobado que es razonable suponer que el valor de V no cambia en la zona derecha del dominio de \tilde{Q} , siempre que Q_{max} se tome lo suficientemente grande (a partir de 6 veces \bar{Q}_i). Por ello, y tras descartar condiciones de contorno tipo Dirichlet (que imponen un valor a V que no se puede conocer, por lo que resultan artificiales y conllevan una resolución incorrecta de la ecuación) se selecciona una condición de contorno tipo Neumann homogénea para el extremo derecho del dominio de \tilde{Q} . Es decir, se especifica que

$$\frac{\partial V}{\partial \tilde{Q}}(1, t) = 0 \quad (4.8)$$

y por tanto la representación del valor de V en el extremo derecho del dominio de \tilde{Q} será horizontal.

4.4. Discretización en tiempo: método de las características

Para aproximar la solución del problema de derivadas parciales dado por la ecuación (4.6) junto con las condiciones (4.7) y (4.8) se propone la discretización en tiempo mediante el método de las características

y en la variable demanda mediante elementos finitos. Dicha aproximación ha sido utilizada con éxito en trabajos anteriores de valoración de opciones vanilla europeas y americanas en [13], bonos con recompra en [3], y en opciones asiáticas en [1]. En el caso del presente problema, en primer lugar se introducen los instantes de tiempo discreto. Para ello, dado $M > 1$, se considera el conjunto de tiempos t^0, t^1, \dots, t^M y la notación

$$f^n(\tilde{Q}) = f(\tilde{Q}, t^n), \quad n = 0, 1, \dots, M, \quad \tilde{Q} \in [0, 1]$$

donde

$$t^n = n\Delta t$$

Por otra parte, se define la derivada total

$$\frac{DV}{Dt}(\tilde{Q}, t) = \frac{dV}{dt}(\tilde{Q}(t), t) = \frac{\partial V}{\partial \tilde{Q}}(\tilde{Q}, t) \frac{d\tilde{Q}}{dt}(t) + \frac{\partial V}{\partial t}(\tilde{Q}, t), \quad (4.9)$$

que se puede escribir en la forma:

$$\frac{DV}{D\tilde{Q}}(\tilde{Q}, t) = \frac{\partial V}{\partial \tilde{Q}}(\tilde{Q}, t)[(\mu - \sigma k - \sigma^2)\tilde{Q}] + \frac{\partial V}{\partial t}(\tilde{Q}, t) \quad (4.10)$$

donde:

$$\frac{d\tilde{Q}}{dt}(t) = (\mu - \sigma k - \sigma^2)\tilde{Q} \quad (4.11)$$

Con esta notación, la ecuación (4.6) se escribe de la forma:

$$\frac{DV}{Dt} + \frac{\partial}{\partial \tilde{Q}} \left(\frac{\sigma^2}{2} \tilde{Q}^2 \frac{\partial V}{\partial \tilde{Q}} \right) - rV + R = 0 \quad (4.12)$$

la discretización temporal se basa en la aproximación de la derivada total en el punto (\tilde{Q}, t^{n+1}) en la forma:

$$\frac{DV}{Dt}(\tilde{Q}, t^{n+1}) = \frac{V(\chi^{n+1}(\tilde{Q}), t^{n+1}) - V(\tilde{Q}, t^n)}{\Delta t} \quad (4.13)$$

donde $\chi^{n+1}(\tilde{Q}) = \chi(t^{n+1})$, siendo $\chi(\tau)$ la solución del problema de valor inicial:

$$\chi'(\tau) = (\mu - \sigma k - \sigma^2)\chi(\tau), \quad \chi(t^n) = \tilde{Q} \quad (4.14)$$

Para resolver dicho problema se utiliza la aproximación por el método de Euler con paso Δt :

$$\chi^{n+1}(\tilde{Q}) = \tilde{Q} + \Delta t(\mu - \sigma k - \sigma^2)\tilde{Q} \quad (4.15)$$

Sustituyendo (4.13) en (4.12) se plantea en cada paso del tiempo el problema aproximado para calcular V^n :

$$\frac{V^{n+1} \circ \chi^{n+1} - V^n}{\Delta t} + \frac{d}{d\tilde{Q}} \left(\frac{\sigma^2}{2} \tilde{Q}^2 \frac{dV^n}{d\tilde{Q}} \right) - rV^n + R = 0 \quad (4.16)$$

A continuación, es necesario obtener condiciones de contorno adecuadas para este problema a partir de (4.7) y (4.8) asociadas al problema de partida.

En el caso de la condición (4.7) se realiza la aproximación por diferencias finitas:

$$\frac{\partial V}{\partial t}(0, t^n) \simeq \frac{V(0, t^{n+1}) - V(0, t^n)}{t^{n+1} - t^n}$$

o:

$$\frac{dV^n}{dt}(0) \simeq \frac{V^{n+1}(0) - V^n(0)}{t^{n+1} - t^n}$$

que, sustituyendo en (4.7) conduce a la condición de Dirichlet para (4.16) en $\tilde{Q} = 0$:

$$V^n(0) = \frac{V^{n+1}(0)}{1 + (t^{n+1} - t^n)r} = V_0^n \quad (4.17)$$

De modo más sencillo, la condición (4.8) se traduce en la condición Neumann homogénea para (4.16) en $\tilde{Q} = 1$:

$$\frac{dV^n}{d\tilde{Q}}(1) = 0 \quad (4.18)$$

El algoritmo en la variable tiempo consiste en calcular V^n a partir de V^{n+1} , partiendo de $n = M - 1$ hasta $n = 0$. Una vez planteadas las condiciones de contorno, con vistas a la resolución por elementos finitos, se introduce la formulación variacional asociada al problema definido por (4.16), (4.17) y (4.18).

Dados los espacios funcionales:

$$V_0 = \{\varphi \in H^1(0, 1), \varphi(0) = 0\}$$

$$V_1^n = \{\varphi \in H^1(0, 1), \varphi(0) = V_0^n\}$$

Se plantea la formulación variacional:

Encontrar $V^n \in V_1^n$ tal que:

$$\int_0^1 \frac{V^{n+1} \circ \chi^{n+1} - V^n}{\Delta t} \varphi - \int_0^1 \frac{\sigma^2}{2} \tilde{Q}^2 \frac{dV^n}{d\tilde{Q}} \frac{d\varphi}{d\tilde{Q}} - \int_0^1 rV^n \varphi + \int_0^1 \bar{R}^n \varphi = 0 \quad \forall \varphi \in V_0 \quad (4.19)$$

o, equivalentemente (multiplicando por Δt y agrupando términos):

$$(1 + \Delta t r) \int_0^1 V^n \varphi + \frac{\Delta t \sigma^2}{2} \int_0^1 \tilde{Q}^2 \frac{dV^n}{d\tilde{Q}} \frac{d\varphi}{d\tilde{Q}} = \int_0^1 (V^{n+1} \circ \chi^{n+1}) \varphi + \Delta t \int_0^1 \bar{R}^n \varphi \quad \forall \varphi \in V_0 \quad (4.20)$$

4.4.1. Discretización por elementos finitos

Para aproximar la formulación variacional de (4.20), se utiliza el espacio de elementos finitos:

$$V_h = \{\varphi_h \in \zeta[0, 1] \mid \varphi_h|_{[\tilde{Q}_j, \tilde{Q}_{j+1}]} \text{ es un polinomio de grado } \leq 1\}$$

asociado a una malla de elementos finitos del intervalo $[0, 1]$, con nodos, no necesariamente igualmente repartidos,

$$\{\tilde{Q}_0, \dots, \tilde{Q}_n\}$$

Cada elemento finito es un subintervalo $[\tilde{Q}_j, \tilde{Q}_{j+1}]$. Dados

$$V_{h,0} = \{\varphi_h \in V_h \mid \varphi_h(0) = 0\}$$

$$V_{h,1}^n = \{\varphi_h \in V_h \mid \varphi_h(0) = V_0^n\}$$

el problema discretizado por elementos finitos resultante es:

Encontrar $V_h^n \in V_{h,1}^n$ verificando:

$$(1 + \Delta t r) \int_0^1 V_h^n \varphi_h + \frac{\Delta t \sigma^2}{2} \int_0^1 \tilde{Q}^2 \frac{dV_h^n}{d\tilde{Q}} \frac{d\varphi_h}{d\tilde{Q}} = \int_0^1 (V_h^{n+1} \circ \chi^{n+1}) \varphi_h + \Delta t \int_0^1 \bar{R}^n \varphi_h \quad \forall \varphi_h \in V_{h,0}$$

Se divide cada una de las integrales en la suma de las integrales en cada uno de los elementos finitos:

$$\sum_{j=0}^N (1 + \Delta t r) \int_{\tilde{q}_j}^{\tilde{q}_{j+1}} V_h^n \varphi_h + \sum_{j=0}^N \frac{\Delta t \sigma^2}{2} \int_{\tilde{q}_j}^{\tilde{q}_{j+1}} \tilde{Q}^2 \frac{dV_h^n}{d\tilde{Q}} \frac{d\varphi_h}{d\tilde{Q}} \simeq \sum_{j=0}^N \int_{\tilde{q}_j}^{\tilde{q}_{j+1}} (V^{n+1} \circ \chi^{n+1}) \varphi_h + \sum_{j=0}^N \Delta t \int_{\tilde{q}_j}^{\tilde{q}_{j+1}} \bar{R}^n \varphi_h$$

A partir de aquí, cada uno de estos elementos finitos dará lugar a una de las submatrices elementales que se ensamblarán para calcular la matriz elemental y el segundo miembro elemental del método de elementos finitos. En concreto, para este problema las submatrices elementales tienen la siguiente expresión:

$$[A_h^j] = (1 + \Delta t r) [A_h^j]^1 + \frac{\Delta t \sigma^2}{2} [A_h^j]^2 \quad (4.21)$$

donde la matriz $[A_h^j]^1$ se puede calcular de forma analítica y toma el valor

$$[A_h^j]^1 = (\tilde{Q}_{j+1} - \tilde{Q}_j) \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}, \quad (4.22)$$

y la matriz $[A_h^j]^2$ se aproxima mediante el método de trapecio, resultando un valor de

$$[A_h^j]^2 = \frac{(\tilde{Q}_{j+1}^2 - \tilde{Q}_j^2)}{2(\tilde{Q}_{j+1} - \tilde{Q}_j)} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \quad (4.23)$$

En cuanto al segundo término elemental, cada uno de los subvectores se calcula como

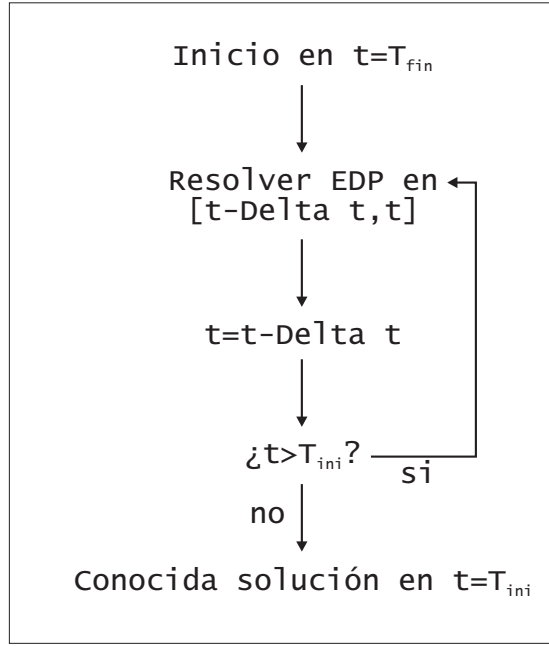


Figura 4.1: Proceso de resolución de la ecuación en derivadas parciales

$$(b_h^j) = (b_h^j)^1 + \Delta t (b_h^j)^2, \quad (4.24)$$

donde el valor de $(b_h^j)^1$ se aproxima como

$$(b_h^j)^1 = (\tilde{Q}_{j+1} - \tilde{Q}_j) (V^{n+1} \circ \chi^{n+1}) \left(\frac{(\tilde{Q}_{j+1} + \tilde{Q}_j)}{2} \right) \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad (4.25)$$

con

$$(V^{n+1} \circ \chi^{n+1})(\tilde{Q}) = V^{n+1}(1 + \Delta t(\mu - \sigma k - \sigma^2)\tilde{Q}) \quad (4.26)$$

Por último, la expresión de $(b_h^j)^2$ se aproxima mediante el método de trapecio

$$(b_h^j)^2 = (\tilde{Q}_{j+1} - \tilde{Q}_j) \begin{pmatrix} \frac{\bar{R}^n(\tilde{Q}_j)}{2} \\ \frac{\bar{R}^n(\tilde{Q}_{j+1})}{2} \end{pmatrix} \quad (4.27)$$

En la Figura 4.1 se muestra el proceso general de resolución de la ecuación parabólica mediante el método explicado en este capítulo.

De este modo se han calculado todas las expresiones necesarias para la aplicación del método de elementos finitos con características a la ecuación en derivadas parciales parabólica que aparece en el problema de valoración de inversiones. Con este método se puede calcular una aproximación de la solución en el intervalo seleccionado; esta resolución se utilizará en los algoritmos de toma de decisiones de las redes de comunicaciones, como se mostró en los capítulos dedicados a los modelos matemáticos.

Capítulo 5

Resolución distribuida del problema

En el presente capítulo se realiza un análisis sobre la paralelización de los dos problemas estudiados, para averiguar si se puede reducir el tiempo de computación empleado en el cálculo de tomas de decisión, mediante el uso de varios ordenadores de propósito general, como los que pueden ser encontrados en cualquier empresa mediana o grande. Este análisis está recomendado, ya que los problemas implicados (sobre todo el de valoración de redes de telefonía móvil) tienen una elevada carga computacional, al tener que resolver muchas veces varias ecuaciones en derivadas parciales, y cualquier grado de paralelización puede ser aprovechado para reducir los tiempos de cálculo. En concreto, la paralelización que se busca es de alto nivel, para ser explotada mediante una red de ordenadores de propósito general comunicados mediante una red no especializada; esto hace que, al ser la comunicación más lenta que en hardware específicamente diseñado o redes de alta velocidad, las tareas que se resuelvan en paralelo deben ser lo más grande posibles y realizar el mayor trabajo posible, para minimizar el impacto de las comunicaciones por la red.

5.1. Resolución de una EDP general

En este Proyecto no se aborda la paralelización del método de elementos finitos, que no resulta sencilla. Como se vio en el capítulo anterior, la resolución de la ecuación en derivadas parciales parabólica (4.1) se realiza resolviendo secuencialmente una serie de ecuaciones elípticas; la solución de cada una de esas ecuaciones se obtiene partiendo de la solución de la ecuación anterior (la siguiente en el tiempo, ya que se resuelven hacia atrás). Esto hace que aparezca una dependencia de datos entre las partes del programa que resuelven cada una de las ecuaciones de una dimensión, por lo que la resolución de las mismas se tiene que realizar en serie. La otra fuente de paralelismo que aparece en el método de elementos finitos es en el propio método aplicado a cada una de las ecuaciones de una dimensión; este método está dividido en dos partes, como todos los métodos de elementos finitos:

- Ensamblado de la matriz elemental: es el primer paso del método, en el cual se calculan las submatrices elementales de tamaño 2×2 y se ensamblan en la matriz elemental, y los subvectores elementales de tamaño 2×1 y se ensamblan en el segundo miembro elemental. Cada una de estas submatrices elementales se puede calcular independientemente, ya que solamente dependen del elemento finito que se esté considerando (observense las ecuaciones 4.21, 4.22, 4.23, 4.24, 4.25, 4.26 y 4.27). De esta forma, se pueden calcular el conjunto de matrices de forma separada, pero no se pueden ensamblar de forma paralela todas, ya que al ensamblar dos matrices contiguas el elemento (2, 2) de la primera matriz se solapa con el elemento (1, 1) de la segunda, por lo que aparece una dependencia de datos (concretamente una dependencia de salida, ya que ambas matrices modifican el mismo elemento de la matriz elemental). Sin embargo, las matrices elementales se pueden ensamblar en dos pasos; primero las pares de forma paralela, ya que las matrices pares no se solapan, y luego las impares, que tampoco se solapan (solamente se solapan las matrices consecutivas). El mismo razonamiento se puede aplicar al ensamblado de los subvectores elementales, ensamblando primero los pares y luego los impares. Así, existe una fuente de paralelismo en el cálculo de la matriz elemental y el segundo miembro elemental del método de elementos finitos.
- Resolución del sistema de ecuaciones: la matriz y el segundo miembro elementales forman un sistema de ecuaciones tridiagonal, que se resuelve mediante el método LU para matrices tridiagonales. Este método realiza tres pasadas secuenciales sobre cada una de las diagonales, utilizando para el cálculo de cada elemento de dichos vectores el elemento calculado en la iteración anterior; esta dependencia de datos hace que no se pueda resolver de forma paralela el sistema de ecuaciones.

Se ha mostrado que la única fuente de paralelismo que hay en el método de elementos finitos es en el cálculo y ensamblado de la matriz y segundo miembro elemental. Sin embargo, este paralelismo no va a ser explotado, porque es de grano muy fino para el propósito buscado:

- El cálculo de cada submatriz elemental requiere la suma de dos matrices 2×2 , multiplicadas por unos escalares calculables en tres o cuatro instrucciones. Es decir, el cálculo es muy sencillo para cada una de las matrices, y los datos que necesitan son $\tilde{q}_j, \tilde{q}_{j+1}, \Delta t, r, y, \sigma$.
- El cálculo del segundo miembro elemental requiere menos operaciones, ya que los vectores son de tamaño 2×1 , Sin embargo, los datos que se necesitan son mucho mayores, ya que para calcular el vector $(b_h^j)^1$ se necesita la solución entera de la ecuación diferencial anterior, que puede tener desde cientos de elementos a decenas o cientos de miles. Esto hace que no sea razonable el calcularlos de forma paralela, ya que habrá que enviar por la red una gran cantidad de información para realizar unas operaciones muy sencillas.

Por lo tanto, el cálculo que puede ser realmente explotado es el de las submatrices elementales, ya que no se puede distribuir de forma eficiente toda la información necesaria para calcular los subvectores

elementales. Sin embargo, para la aplicación a desarrollar este es un grano de paralelismo demasiado fino, ya que no es rentable computacionalmente el esperar al cálculo remoto de las matrices 2×2 utilizando una red no especializada y ordenadores de propósito general. Por ello, este paralelismo no se utilizará, y se resolverá el método de elementos finitos con un único procesador.

5.2. Redes de fibra óptica

El algoritmo de valoración de redes de fibra óptica para acceso a Internet se puede dividir en dos grupos de tareas, las de cálculo de valores y las de toma de decisiones. Las primeras toman como entrada un intervalo de tiempo y la ecuación de una línea y resuelven dicha ecuación desde la fecha final del intervalo hasta la fecha inicial (realizando los pagos de mantenimiento adecuados en ese intervalo). Las segundas toman el valor de una serie de líneas (de una línea y de todas las posibilidades de mejora de dicha línea) y actualizan el valor de la línea almacenando las decisiones tomadas. De los dos grupos, claramente el que mayor coste computacional tiene es el primero, ya que aplicar sucesivamente el método de elementos finitos es mucho más costoso que comparar una serie de valores; por esta razón, son estas tareas las que se van a ejecutar de forma distribuida. Las dependencias entre estas tareas se muestran en la Figura 5.1. En esa figura se muestran en blanco las tareas del grupo 1 (cálculo de valores) y en gris las del grupo 2 (toma de decisiones).

Se aprecia que el algoritmo se divide en varias *calles*, una para cada línea. En la figura se muestran cuatro calles: con flechas azules la calle de la última línea, con flechas rojas la calle de la penúltima, con flechas negras la calle de la segunda línea y con flechas verdes la de la primera línea. Las dependencias entre tareas muestran que las tareas de una calle solamente dependen de las tareas de esa misma calle en tiempos mayores y de las tareas de calles de líneas de mayor capacidad. Es decir, la calle de la última línea (azul) no depende de ninguna otra calle, y sus tareas se han de realizar de forma secuencial. Sin embargo, las tareas de la penúltima calle dependen de las tareas de la última calle; en las zonas de tipo II, para calcular el valor de la línea $n - 1$ mejorada a la línea n es necesario calcular el valor de la línea n en el final de la zona, y ese cálculo se realiza en la calle azul. De la misma forma, para calcular los valores de una línea a los tipos mejores que ella es necesario conocer el valor de esos tipos en la fecha de mejora, ya que son el valor final de la línea mejorada a esos tipos. Así, al calcular el valor de una línea j en el principio de una zona de tipo II (o sea, cuando se ha completado el algoritmo de decisión para una línea) se puede comenzar a calcular el valor de dicha línea en la zona I anterior (si la hubiese). Este cálculo sólo depende del algoritmo de decisión anterior. En cambio, cuando se calcula el valor de esa línea al principio de la zona de tipo I se pueden calcular $j + 1$ valores, el valor de la línea j en la zona de tipo II anterior y el valor de las j líneas de menor capacidad que la j mejoradas a la línea j en la misma zona II. Esto es una fuente de paralelismo a alto nivel, ya que se puede ir resolviendo las calles sincronizándolas

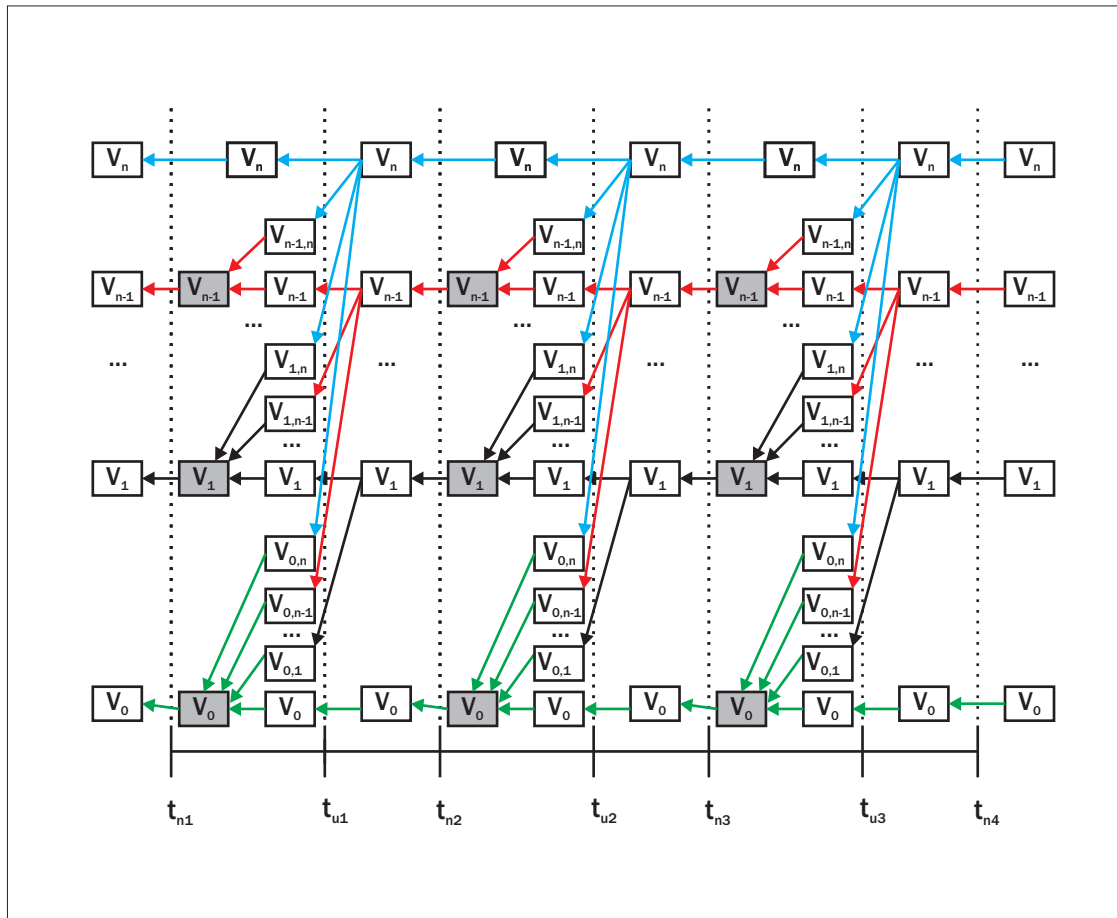


Figura 5.1: Dependencias entre tareas del algoritmo de redes de fibra óptica

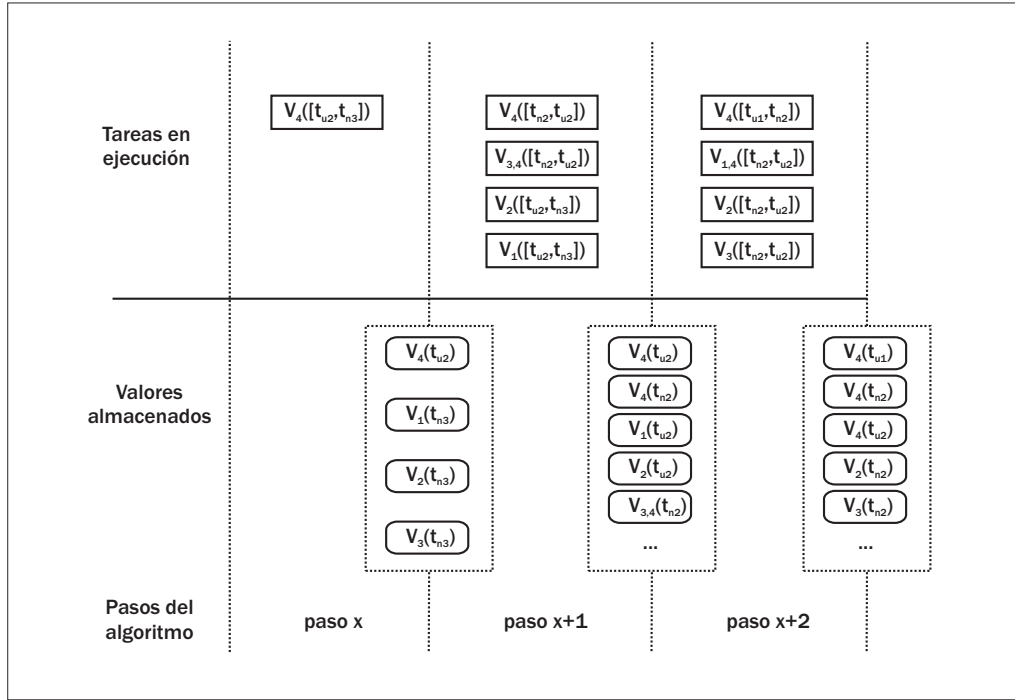


Figura 5.2: Posible escenario donde se almacena el valor de la línea 4 en diferentes t al final de un paso del algoritmo

solamente con las calles de mayor capacidad.

Sin embargo, el realizar todas las tareas en el momento en el que pueden realizarse por las restricciones impuestas entre tareas provoca que haya que guardar el valor de una línea en varios momentos simultáneamente. Por ejemplo, si se calcula el valor de la línea 4 en el principio de una zona I, se podrán calcular 5 valores simultáneamente, el nuevo valor de la línea 4 y los valores de las líneas 0, 1, 2 y 3 mejoradas a la línea 4. Una vez calculado el primero de los valores (el valor de la línea 4 al principio de la zona II), se podrá calcular el valor de esa línea en la anterior zona I, y así sucesivamente. Esto hace que se tengan que almacenar al mismo tiempo los valores de la línea 4 en diferentes momentos (ver Figura 5.2). Si esta situación se repite para varias de las líneas, y no se imponen restricciones adicionales, si no se dispone de suficiente hardware para realizar todas las tareas posibles (lo cual es algo normal si el número de líneas es alto) y las tareas a ejecutar en cada paso del algoritmo se seleccionan de forma no razonable, se almacenarán muchos valores de la misma línea en diferentes momentos.

Quizá esta redundancia no parezca un problema, pero hay que tener en cuenta que cada valor de una línea consiste en un vector de elementos, cuya longitud depende de la densidad de la malla de elementos finitos que se toma en la variable Q . Así, es muy recomendable reducir al máximo el número de valores que se almacenan al mismo tiempo, ya que cada valor puede contener desde cientos de elementos flotantes en doble precisión a decenas o cientos de miles, dependiendo de la precisión que se quiera obtener en el

cálculo.

Este problema de memoria se puede mitigar imponiendo una restricción de sincronización entre las tareas. La restricción necesaria es la de ejecutar de forma paralela tareas que pertenezcan a una misma zona. Es decir, si no se pueden ejecutar todas las tareas disponibles en un momento por falta de hardware y hay que serializarlas, se elegirán las tareas que calculan valores dentro de esa zona; además, aunque se disponga de suficiente hardware para resolver todas las tareas de una zona y otras de zonas anteriores, solamente se resolverán los subproblemas pertenecientes a la zona más tardía. Con esta restricción se consigue que para cada línea solamente se almacene un único valor; por lo tanto, en las zonas de tipo I habrá que almacenar n valores para n líneas, y en las zonas de tipo II habrá que almacenar $\frac{n(n+1)}{2}$ valores, que será el máximo del algoritmo. Así, cada valor es almacenado el menor tiempo posible como media, y por ello se minimiza el número de valores que se tiene que mantener en memoria al mismo tiempo; sin embargo, se mantiene una gran cantidad de paralelismo: se pueden ejecutar simultáneamente n ó $\frac{n(n+1)}{2}$ tareas, dependiendo de la zona en la que se esté resolviendo. En realidad, suponiendo que se dispone de hardware ilimitado y que no hay retardos de comunicación (suposiciones útiles en la teoría, pero absurdas en un entorno real), la versión paralela se ejecutaría en el mismo tiempo, independientemente del número de líneas que se consideren: el paralelismo elimina la dependencia del tiempo de cálculo del número de líneas presentes. Sin embargo, es de esperar que en una implementación real los costes de comunicación hagan que sea necesario un número de líneas, una densidad de malla y un número de ordenadores mínimos para que merezca la pena la resolución distribuida del problema.

5.3. Redes de telefonía móvil

La situación que se presenta en el problema de la toma de decisiones para redes de telefonía móvil es similar a la anterior, pero se puede resolver de forma distribuida de forma más sencilla que en el caso de las redes de fibra óptica. Como en ese caso, las tareas existentes se pueden clasificar en tareas de cálculo de valores (en las cuales se aplica el método de elementos finitos) y las tareas de toma de decisión (en las cuales se comparan los valores calculados en las otras tareas para tomar decisiones de mejora). Las tareas de toma de decisión son menos costosas computacionalmente hablando que las tareas de cálculo de valores; así, en el algoritmo se calcularán de forma distribuida las tareas de cálculo, y las tareas de decisión se realizarán en el procesador original.

El algoritmo de redes de telefonía se compone de un bucle que recorre las fechas de observación t_{obs} desde la última hasta la primera. En cada una de las fechas se actualizan los valores calculados de una forma dependiente del tipo de fecha que sea (ver Sección 3.2.3). Entre dos fechas de observación consecutivas se calculan los valores de los clústers que se estén observando en ese intervalo. Estos valores dependerán de varios factores: el tipo de zona a la que pertenezca el intervalo, el número de clústers del

algoritmo, el grado de solapamiento entre decisiones, etc.

La resolución distribuída se realizará en las tareas que calculan el valor de una línea entre dos fechas de observación. Así, se calcularán de forma simultánea todos los valores que el hardware permita, pero imponiendo la misma restricción que en el caso de las redes de fibra óptica; solamente se calcularán valores de un intervalo simultáneamente. De esta forma, las tareas de un intervalo sólo se ejecutarán cuando hayan terminado las tareas del intervalo siguiente (hablando en términos del tiempo físico del problema). Esto hace que el consumo de memoria se minimice, alcanzándose un buen balance entre uso de memoria y resolución distribuída; es más, al tomar varias decisiones simultáneamente el número de valores que se calculan en cada intervalo es mayor que en el caso de las redes de fibra óptica, obteniéndose un mayor beneficio en tiempo de computación si se dispone de suficientes ordenadores para resolver el problema.

Parte III

Implementacion de la aplicación

Capítulo 6

Desarrollo de la aplicación

En este capítulo se presenta el proceso de desarrollo seguido para la realización de la aplicación objetivo. Los pasos mostrados se muestran en el orden seguido en el tiempo a la hora de la construcción de la aplicación: desarrollo de los prototipos para el método de elementos finitos, para los dos algoritmos de toma de decisiones, el desarrollo del sistema utilizado para el cálculo distribuido y la construcción de la aplicación final.

6.1. Método de elementos finitos

El primer paso a la hora de construir la aplicación es el desarrollo iterativo de un prototipo para comprobar la resolución correcta de la ecuación en derivadas parciales parabólica mediante el método de elementos finitos con características explicado en el capítulo 4. Para el desarrollo de ese prototipo se eligió como lenguaje de implementación Matlab, ya que proporciona un entorno amigable para el uso de matrices, resolución de sistemas de ecuaciones y visualización gráfica de resultados, tanto en dos dimensiones como en tres (algo necesario, ya que la representación de la solución de la ecuación en derivadas parciales tiene que realizarse en tres dimensiones).

A partir del desarrollo teórico del método se construye el algoritmo de dicho método, en los dos pasos habituales para los métodos de elementos finitos: construcción del sistema de ecuaciones del método y resolución de dicho sistema. El prototipo desarrollado hace incapié en la resolución del método con mallas no uniformes en la variable Q , para poder ajustar dichas mallas para una resolución óptima del problema.

6.1.1. Validación del prototipo

Una vez construido el prototipo, se han de validar los resultados; el problema a la hora de la validación de dichos resultados para este método de elementos finitos es la no existencia de solución exacta de la ecuación resuelta. Esto hace necesario una forma de validación diferente a la comparación de resultados

entre la teoría y la práctica. Este método se basa en la reducción de la ecuación del problema a otra ecuación con solución conocida, y la incorporación posterior de las características eliminadas en dicha reducción, comprobando que los resultados se mantienen en los límites esperados. La ecuación elegida es la ecuación de valoración de opciones europeas de tipo lookback. Esta ecuación sí posee una solución exacta y conocida, pero es diferente de la ecuación de valoración líneas de comunicación en varios puntos (las condiciones de contorno, tanto en T como en los extremos de Q , la función de beneficio...). El proceso seguido exactamente es el siguiente:

- Reducción de la ecuación: la ecuación 4.1 se convierte en la ecuación de valoración de opciones lookback europeas, y se comprueba que los resultados obtenidos por el método desarrollado concuerdan con la solución teórica.
- Incorporación del valor final ($t = T$): el siguiente paso es incorporar el valor final de la línea o clúster que se utiliza en el presente problema; al ser diferente que el valor utilizado en las opciones europeas, no se pueden comparar los resultados de ambas ecuaciones directamente; para considerar válidos los resultados con el nuevo valor final, se observan dos cosas: el mantenimiento de la convergencia del método y la aparición de resultados razonables. La convergencia es vital a la hora de resolver una ecuación mediante un método numérico, por lo que al aumentar sucesivamente la precisión del método (en este caso, mediante el refinado de la malla de elementos finitos) se tienen que obtener aproximaciones cada vez más cercanas a la solución real, y dichas aproximaciones no pueden oscilar de un lado al otro de la solución real. En cuanto a los resultados razonables, se tiene que mantener la forma de la solución al cambiar el valor final; por ejemplo la diferencia entre dos cálculos con valores finales constantes pero diferentes, debería ser cuantitativa, no cualitativa. Al incorporar el valor final realista se observa que la convergencia se mantiene, y los resultados se mantienen dentro de límites perfectamente predecibles, así que se considera correcto.
- Incorporación de condiciones de contorno: el siguiente paso a considerar son las condiciones de contorno en los extremos de Q , ya que de nuevo son diferentes a las de las opciones lookback europeas (ver sección 4.3 para recordar la derivación de ambas condiciones de contorno). Tras cambiarlas y realizar los cálculos, los resultados obtenidos superan ambas pruebas, de convergencia y conservación de los resultados.
- Parámetros reales: los parámetros utilizados en las opciones europeas son diferentes a los utilizados en los problemas a tratar, a saber, los parámetros $r, k, \sigma, \mu, Q_{max}, T$. Las pruebas de validación se superan con los nuevos parámetros.
- Función de beneficio: la función de beneficio es la mayor diferencia entre las ecuaciones que se están comparando, ya que es un término que no aparecía en las valoración de opciones lookback europeas.

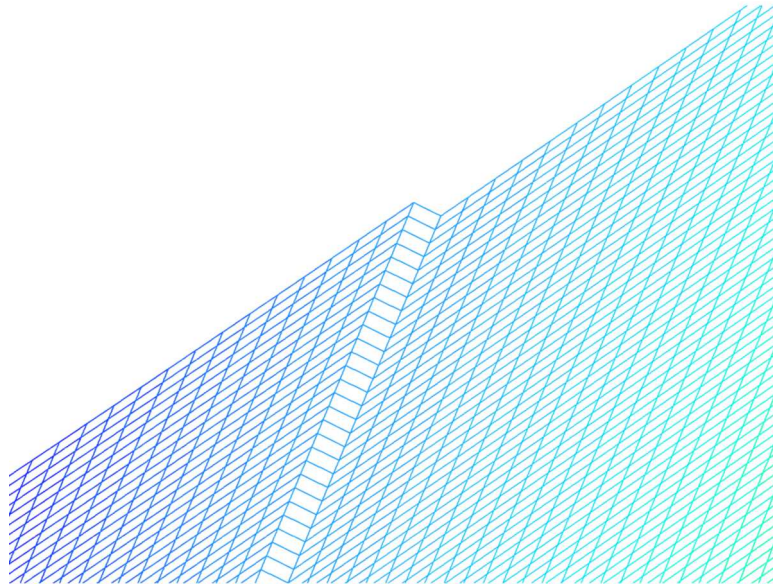


Figura 6.1: Aparición de discontinuidades debidas a los pagos de plazos de mantenimiento

Así, en este paso se incorpora el término R y de nuevo se observa que la convergencia se mantiene y los resultados muestran la evolución predecible al introducir un valor de beneficio dependiente del tiempo y de la demanda de ancho de banda. Con este paso se completa la transición de la ecuación de opciones europeas a la ecuación de valoración de infraestructura de comunicaciones, considerándose correcta la resolución de dicha ecuación.

- Por último, se comprueba la influencia de la inclusión de pagos de mantenimiento en la resolución de la ecuación. Como ya se ha comentado anteriormente, los pagos de mantenimiento son actualizaciones del valor de la línea o clúster en un momento determinado. Estas actualizaciones introducen una discontinuidad en el valor calculado, que en condiciones normales debería ser una superficie continua (ver Figura 6.1).

Las discontinuidades afectan negativamente a los métodos numéricos, reduciendo la velocidad de convergencia en el mejor de los casos e invalidando los resultados en el peor de ellos. En este paso se introducen las discontinuidades de mantenimiento, y se observa que reducen la calidad de las aproximaciones, pero dichas aproximaciones siguen convergiendo a un valor final cuando se refina la malla lo suficiente.

Así, tras la validación se tiene una razonable certeza de que el método de elementos finitos desarrollado es correcto y la implementación realizada es válida.

6.1.2. Optimizaciones aplicadas

Por último, se consideran las optimizaciones a aplicar en el método; en efecto, el método empleado consiste en la aplicación sucesiva de varias resoluciones de una ecuación en una dimensión (mediante el método de elementos finitos en una dimensión). Sin embargo, al analizar las expresiones de las submatrices elementales se puede observar que la matriz elemental del método no depende del momento en el que se está resolviendo la ecuación, sino que solamente depende de los parámetros Δt , r , σ y de la malla que se utilice, pero no de la línea o clúster que se esté considerando, ni del momento en el que se esté resolviendo la ecuación (ver las expresiones de las matrices al final de la sección 4.4.1). Así, la matriz elemental del método $[A_h]$ no varía siempre que no se cambie la malla; esta matriz se calcula al principio de la resolución y se evita el cálculo repetido de la misma; esto acelera el cálculo de una forma notable, ya que se pasa de calcular la matriz $\frac{t_{final}-t_{inicial}}{\Delta t}$ veces a una sola vez, y teniendo en cuenta que su tamaño es $(puntos_q)^2$, el requerimiento computacional disminuye apreciablemente.

Esta optimización no se puede aplicar al segundo miembro elemental, ya que los diferentes vectores a ensamblar dependen del instante de tiempo que se considera, de forma que hay que recalcularlos en cada iteración del método. Sin embargo, se puede aplicar una optimización que disminuye todavía más el tiempo de cálculo en este segundo miembro elemental. En efecto, en el cálculo de dicho vector aparece la expresión

$$(V^{n+1} \circ \chi^{n+1})(\tilde{Q}) = V^{n+1}(1 + \Delta t(\mu - \sigma k - \sigma^2)\tilde{Q}) \quad (6.1)$$

Esta expresión incluye la interpolación de la solución en el paso anterior (V^{n+1}) en un punto $\chi^{n+1}(\tilde{Q})$. La expresión de este punto es $1 + \Delta t(\mu - \sigma k - \sigma^2)\tilde{Q}$, que no depende de nada más que de los parámetros fijos del problema ($\mu, \Delta t, \sigma, k$) y del punto al que se aplica, que pertenece a un conjunto fijo de valores (los puntos de la malla). Así, se pueden precalcular los diferentes puntos $\chi^{n+1}(\tilde{Q})$ al principio de la ejecución para no tener que repetir la búsqueda en el vector V^{n+1} en cada elemento finito de cada iteración del método. Calculando estos valores al principio, se obtiene otro aumento de la velocidad de cálculo, aunque esta optimización tiene la desventaja de consumir memoria adicional para almacenar los puntos precalculados. De todas formas, dado que el número de vectores totales que se maneja es alto, el mantener uno más no es un inconveniente apreciable, mientras que el aumento de velocidad sí es perceptible, por lo que se mantiene esta optimización.

Por último, reseñar que la aplicación de estas mejoras no varía los resultados obtenidos, lo que habría invalidado dichas optimizaciones, sino que se obtienen exactamente las mismas aproximaciones que con el método no optimizado.

6.2. Decisiones para redes de acceso a Internet

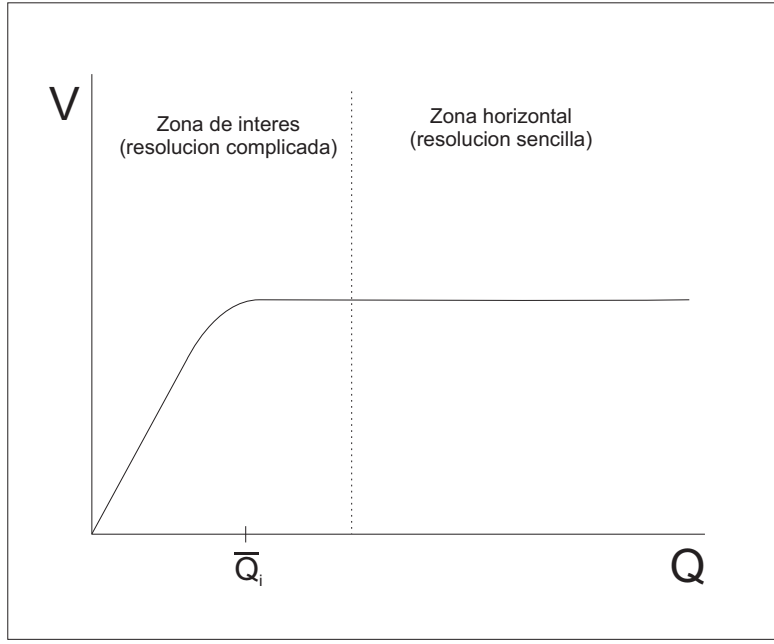
Para implementar el prototipo de toma de decisiones para redes de acceso a Internet se hace uso del prototipo desarrollado para la resolución de las ecuaciones en derivadas parciales mediante el método de elementos finitos con características. Este prototipo se empleará para calcular el valor de cada línea entre cada dos tiempos significativos: el prototipo de redes de fibra óptica añadirá el procesamiento de los valores de las diferentes líneas en cada fecha significativa.

Para realizar dicho procesamiento, incluyendo los pagos de mantenimiento y las tomas de decisión, se aplicarán los algoritmos adecuados de la forma explicada en el capítulo 2. Sin embargo, a la hora de aplicar esos algoritmos surgen tres problemas relacionados entre sí, referentes los tres a la aplicación del método de elementos finitos al conjunto de ecuaciones

- El primer problema es la decisión de usar una malla uniforme (en la que el tamaño de cada elemento finito¹ es igual al del resto), o utilizar una malla no uniforme (en la que el tamaño de cada elemento finito no tiene por qué coincidir con el del resto). En el caso de las ecuaciones que surgen en el problema de valoración de redes de fibra óptica, la mejor opción es la de utilizar una malla no uniforme, debido a la forma de las soluciones a la ecuación en derivadas parciales (mostrada en la Figura 6.2): comenzando en $Q = 0$ y hasta aproximadamente $Q = \bar{Q}_i$ (el ancho de banda máximo de la línea) existe una zona en la que el valor de la solución cambia muy rápidamente; sin embargo, más a la derecha de ese punto (para valores mayores de Q) la solución de la ecuación se estabiliza rápidamente, pasando su gráfica a ser horizontal, sin cambiar más su valor. De esta forma, la zona que mayor precisión requiere a la hora de resolver la ecuación es la parte izquierda, en la que la solución cambia rápidamente, y en especial la zona en la que pasa a ser casi constante. A partir de esa zona la resolución no requiere de tantos puntos, ya que al ser casi constante no importa cuántos puntos se tomen, siempre se aproximará el valor de la solución de una forma similar. Además, la malla de elementos finitos no uniforme debe cambiar suavemente de densidad: si se toma una parte de la malla con una densidad muy grande y otra parte con una densidad mucho menor, sin ningún tipo de transición entre ambas zonas, la solución aparece como una línea quebrada en el punto en el que la malla cambia de densidad, cuando la solución de la ecuación es derivable en todo el dominio de Q . Por esta razón la malla usada tiene que hacer las transiciones entre zonas de diferente densidad de forma progresiva, no de forma brusca.

Teniendo en cuenta estas consideraciones, la malla de elementos finitos entre $Q = 0$ y $Q = Q_{max}$ (más adelante se tratará el valor de dicho Q_{max}) se construye de la siguiente forma: se considera la longitud del primer elemento finito igual a 1, y la longitud del elemento finito n como la longitud del elemento $n - 1$ más 1:

¹un elemento finito es el intervalo comprendido entre dos puntos consecutivos de la malla de elementos finitos

Figura 6.2: Forma de la solución V^n

$$q_{n+1} - q_n = q_n - q_{n-1} + 1$$

Así, cada elemento finito tendrá una longitud mayor que el anterior pero menor que el siguiente, y no habrá transiciones bruscas en ninguna zona de la malla. La longitud total de esta malla será la suma de los N primeros números naturales (si se consideran $N + 1$ puntos, o N elementos finitos). El valor de esta longitud es

$$q_N - q_0 = \sum_{i=1}^N i = \frac{N \times (N + 1)}{2}$$

Esta malla ha de normalizarse para adaptarse al cambio de variable del método de elementos finitos, que considera un dominio $[0, 1]$ en la variable ancho de banda. Para ello, se divide cada punto de la malla por la longitud total de la misma, y así se obtiene una malla no uniforme con transiciones suaves en el dominio $[0, 1]$.

- Otra decisión a tomar surge del hecho de tener que resolver un conjunto de ecuaciones y de tener que comparar dichas soluciones entre sí (para comparar —y actualizar— los valores de las líneas en el algoritmo de toma de decisiones). Al tener un conjunto de ecuaciones, se puede considerar que cada una de ellas tiene su propio dominio: dado que cada línea tiene un ancho de banda máximo, entonces la solución de su ecuación se volverá constante en un punto diferente al del resto de mallas,

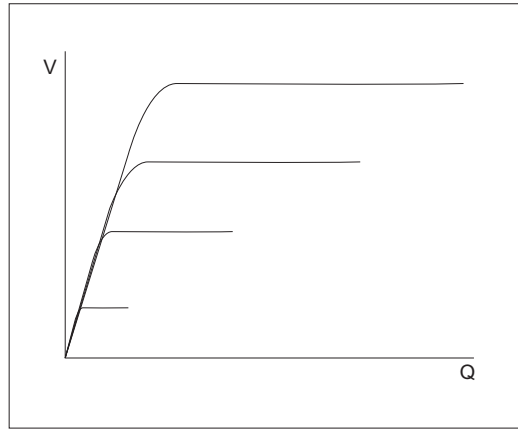


Figura 6.3: Resolución del conjunto de ecuaciones con varios dominios

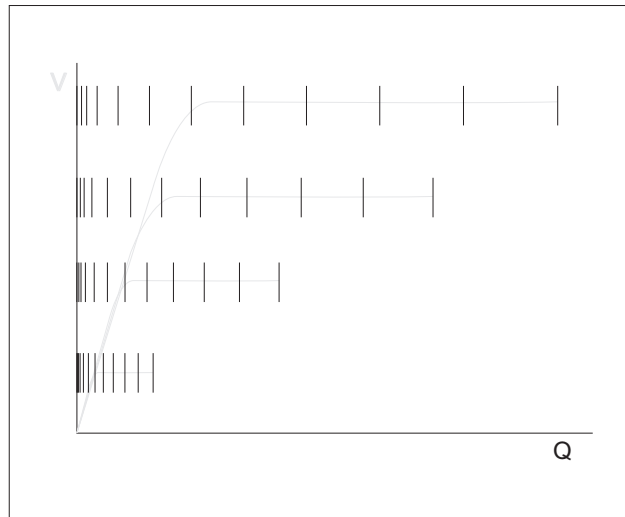


Figura 6.4: Comparación entre las mallas de cada una de las líneas

por lo que solamente hace falta resolver dicha ecuación hasta un punto Q_{max}^i . Cada una de las líneas tendrá su capacidad máxima (ver Figura 6.3).

Así, cada una de las líneas que se quiere evaluar tendrá que tener su propia malla de elementos finitos, que será válida para el dominio de la capacidad de dicha línea. Dado que cada malla está definida en un dominio diferente, los puntos de las diferentes mallas no tienen por qué coincidir con los puntos del resto de mallas (Figura 6.4): las aproximaciones de las diferentes líneas estarán calculadas en diferentes puntos. Sin embargo, para la toma de decisiones es necesario comparar el valor de una línea con el valor del resto de líneas de mayor capacidad. Por supuesto, esta comparación se ha de hacer en el mismo punto de Q . Por esta razón, al usar una malla de elementos finitos independiente para cada una de las líneas surge la necesidad de aplicar interpolación a los resultados obtenidos: cuando sea necesario comparar el valor de la línea i con el resto de líneas en un punto del dominio

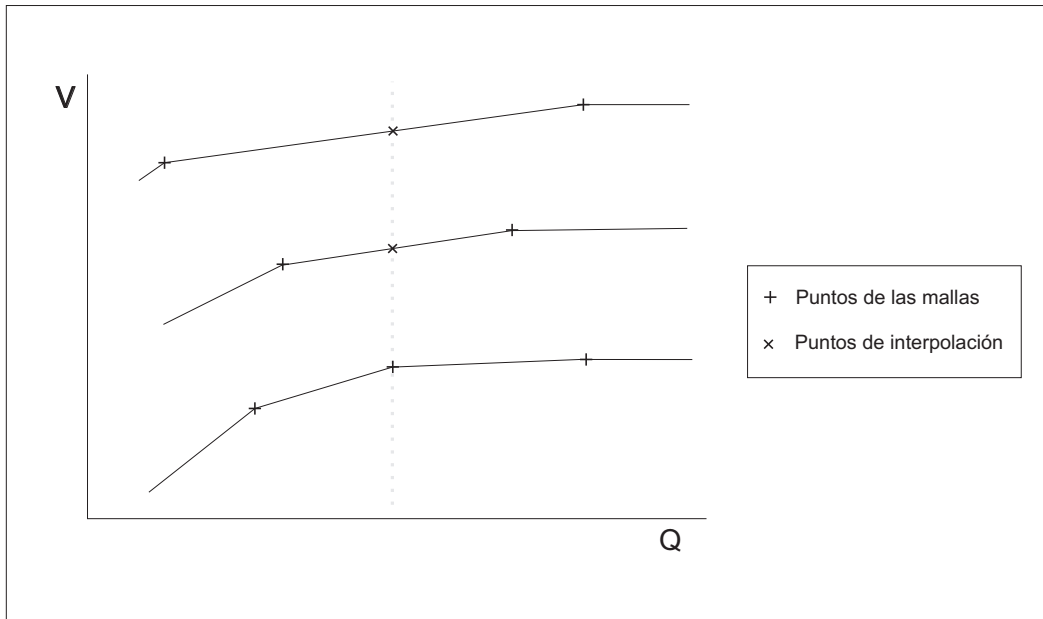


Figura 6.5: Necesidad de interpolación cuando se utilizan varias mallas

de la línea i se interpolará el valor del resto de líneas en dicho punto, a partir de sus valores en los puntos de sus respectivas mallas que estén más próximos al punto de la línea i en el que se quieren comparar todas las líneas (situación mostrada en la Figura 6.5). El aplicar interpolación añade una carga computacional bastante importante cuando se considera que se ha de hacer una comparación para cada una de las líneas, para cada uno de los puntos de Q , para cada una de las líneas de mayor capacidad y para cada una de las fechas de notificación. Sin embargo, dado que cada una de las líneas tiene su propia malla, el número de puntos que se necesita en cada una de éstas es más reducido que si se utiliza una sola malla si se quiere la misma precisión, por lo que ambos efectos se compensan.

La otra opción posible para las mallas, que evita el uso de interpolación, es el uso de una única malla para todas las líneas del problema. Al resolver todas las ecuaciones con la misma malla, está asegurado que las aproximaciones de las soluciones para todas las líneas se obtengan en los mismos puntos, pudiéndose comparar y actualizar de forma inmediata. Obviamente, esta única malla tiene que englobar al dominio de todas las líneas del problema (figura 6.6). El uso de una malla hace que la línea de mayor capacidad se resuelva con la misma precisión que si se utilizan varias mallas, pero las líneas de menos capacidad se resuelven con menor precisión, ya que se resuelven sobre un subconjunto de la malla total, por lo que reciben menos puntos de la misma en las zonas en las que sus soluciones no son constantes. Así, la precisión obtenida para una línea será menor cuanto menor sea su capacidad con respecto a la de la línea de mayor capacidad. Esto obliga a tener que utilizar un mayor número de puntos en la malla de elementos finitos que cuando

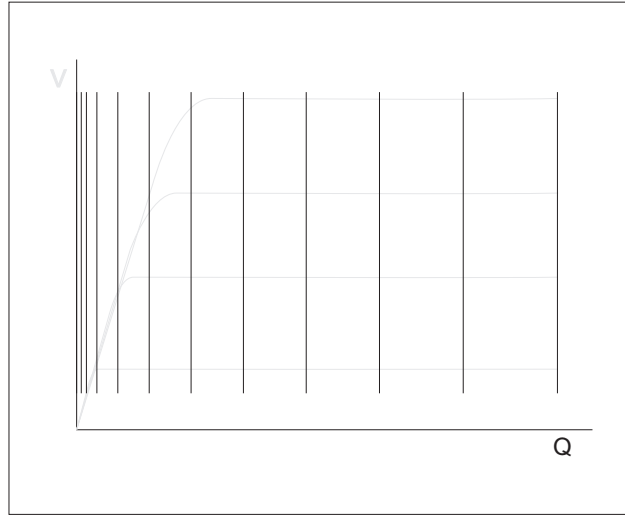


Figura 6.6: Una única malla para la resolución de las diferentes ecuaciones

se resuelve el problema con una malla para cada línea. Además, si se utiliza una sola malla ésta tiene que ser obligatoriamente no uniforme, debido a que el número de puntos en la malla necesario para obtener con la línea de menor capacidad una precisión aceptable es demasiado alto para ser aplicable, tanto en consumo de memoria como en tiempo de cálculo (este número es del orden de millones de puntos en la malla). Con mallas no uniformes el número de puntos se mantiene dentro de límites aceptables, por lo que es la única opción aplicable si se elige el uso de una única malla.

Sin embargo, la opción de una sola malla resuelve un problema que aparece en la resolución del problema con varias mallas, debido a la forma de aplicar el algoritmo de toma de decisiones. Este problema se ilustrará mediante un ejemplo: supónganse tres fechas notables del problema, una fecha de mejora t_{u_1} , la siguiente fecha de notificación t_n y la siguiente fecha de mejora t_{u_2} . Estas tres fechas definen una zona de tipo I (entre las dos primeras) y una zona de tipo II (entre las dos últimas).

Se consideran solamente dos líneas en el ejemplo. Su valor en el momento t_{u_2} se muestra en la Figura 6.7; en dicha figura se compara el valor de ambas líneas en el caso de utilizar una malla común con el caso de usar dos mallas diferentes. Partiendo de este valor final, se resuelven las ecuaciones de ambas líneas hasta la fecha de notificación t_n y se aplica el algoritmo de decisión (Figura 6.8). En esta figura se muestra para cada línea su valor en t_{u_2} y su valor en t_n después de aplicar el algoritmo de decisión, junto con los valores de ambas líneas en la fecha final t_{u_2} , en colores más apagados (estos valores son los que fueron mostrados en la Figura 6.7). Se observa que la línea de menor capacidad no se mejora si se utilizan mallas diferentes para las dos líneas, mientras que sí se actualiza si se utiliza una sola malla, porque el valor de la línea de menor capacidad mejorada a la otra línea sólo supera a su valor sin mejorar para una demanda de capacidad mayor que la

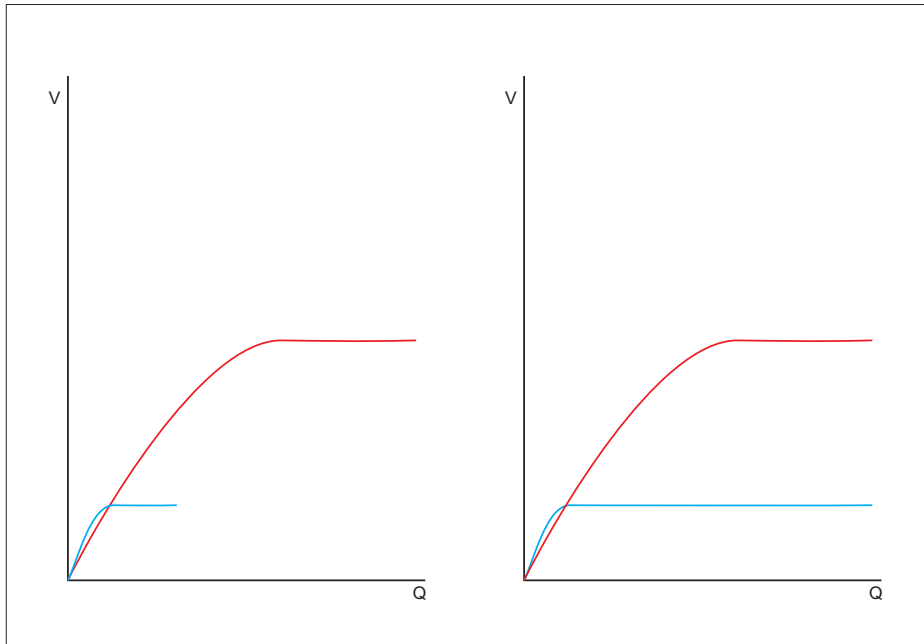


Figura 6.7: El valor de ambas líneas en la fecha final t_{u2}

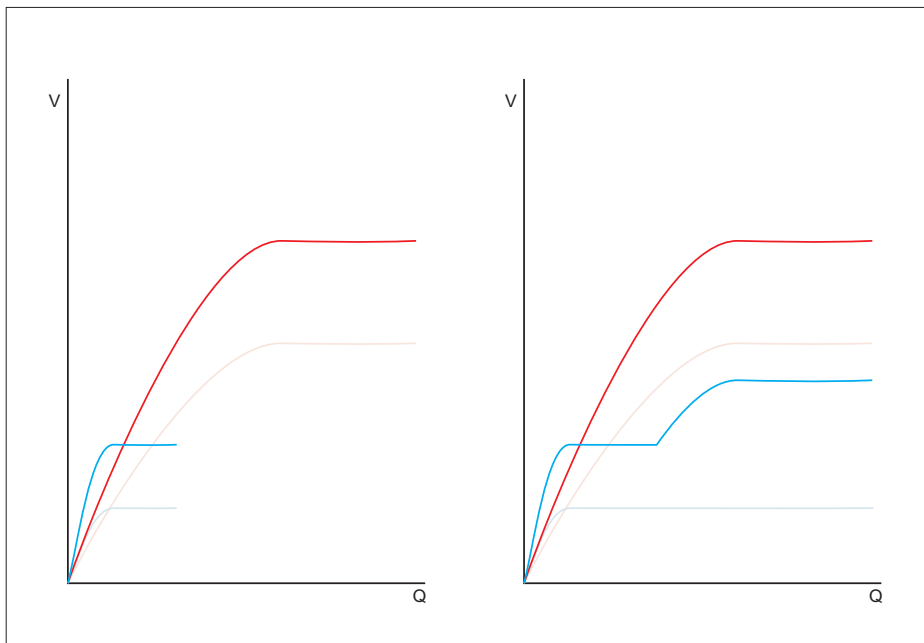


Figura 6.8: El valor de ambas líneas en la fecha de notificación t_n

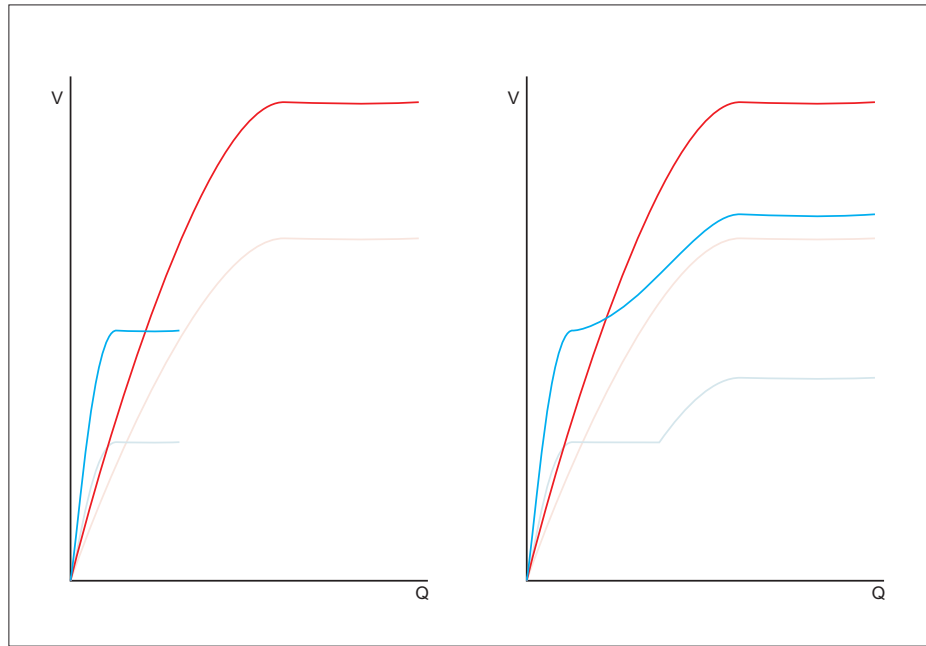


Figura 6.9: El valor de ambas líneas en la fecha de mejora t_{u1}

considerada en la malla más pequeña usada en el otro caso. Puede parecer que al realizarse muy por encima de la capacidad máxima de la línea esta mejora no aporta ninguna diferencia con respecto al valor obtenido con dos mallas. Sin embargo, al resolver las dos ecuaciones en la zona de tipo I, se obtienen valores diferentes para la línea de menor capacidad, dependiendo de si se han calculado con una o dos mallas (ver Figura 6.9).

Es decir, en el caso de usar varias mallas pueden existir actualizaciones fuera del dominio de la malla que si se tuviesen en cuenta afectarían al cálculo de valores anteriores (en tiempo físico) de la línea.

Además de este inconveniente, el uso de diferentes mallas de elementos finitos presenta otro problema, relacionado con la condición de contorno en el extremo derecho del dominio de Q . Si se usa una única malla se toma de tal forma que englobe todas las mallas más pequeñas; en el extremo derecho de la misma la condición impuesta es una condición Neumann homogénea, escogida de forma empírica, pero siendo aún así coherente. En efecto, por muchas operaciones que se realicen, en el punto máximo de Q siempre se ha de cumplir que la solución es horizontal, debido a la forma de las ecuaciones y a que el extremo derecho de la malla estará situado mucho más a la derecha que la máxima de las capacidades máximas a tener en cuenta (ver más adelante). Sin embargo, si se utilizan varias mallas esta situación no tiene por qué cumplirse siempre; al inicio del problema si será válida, pero según se vayan realizando actualizaciones, puede que la solución de una ecuación se actualice con los valores de otra solución que en esos puntos no sea horizontal. A partir de ese

momento, la resolución de la ecuación parte de un valor inicial que no es horizontal en su parte derecha, de forma que la condición de contorno no puede ser del tipo Neumann homogénea. Así, la condición de contorno válida en el extremo derecho cuando se usa una malla no es válida cuando se usan varias.

Por estas dos razones, se descarta el uso de varias mallas de elementos finitos, calculando el valor de todas las líneas sobre la misma malla.

- Por último, es necesario mencionar la decisión que se ha de tomar acerca del valor máximo de ancho de banda para el que se resuelven las ecuaciones. Dado que se va a utilizar una sola malla, el valor máximo de Q que se va a utilizar tiene que ser mayor que la máxima de las capacidades máximas de todas las líneas. Por razones numéricas, concretamente, para que la condición Neumann homogénea no influya en la solución en la zona de interés, la ecuación ha de resolverse como mínimo hasta 6 veces la capacidad máxima de la línea. Por ello, se elige un valor de Q_{max} igual a 6 veces la capacidad máxima de la línea de mayor capacidad, para obtener una mayor estabilidad en las soluciones.

En cuanto a la validación del prototipo, se realiza contra los resultados presentados en [4]. En dicho artículo se presentan una serie de decisiones en forma de porcentaje de capacidad, pero no se hace ninguna mención a los valores de las líneas. Esto hace que la validación del prototipo se tenga que hacer en un solo paso, sin poder comprobar la validez individual de cada parte del mismo, sino que el único punto que se puede comprobar son los resultados finales. Estos resultados se intentan hacer lo más parecidos a los del artículo del que se extrajo el modelo de redes de acceso a Internet; en concreto, se considera válido el prototipo si los resultados son parecidos en valor numérico y además muestran todos los efectos que se comentan en [4].

El lenguaje elegido para la implementación del cálculo de decisiones de redes de fibra óptica es una combinación de Matlab y Java. Matlab se utiliza en las primeras fases del desarrollo del prototipo, en las cuales es útil disponer de las facilidades para visualización de resultados de forma gráfica y manejo de matrices que proporciona Matlab. Sin embargo, es demasiado lento para manejar mallas con un número suficientemente grande de puntos, de forma que las últimas fases del prototipo se desarrollan en lenguaje Java, que realiza los cálculos lo suficientemente rápido, pese a no ser tan rápido como lenguajes compilados como C o Fortran. Sin embargo, el código Java se puede compilar a código nativo mediante algunos compiladores que proporcionan esta opción, obteniéndose velocidades mucho mayores. En el curso del desarrollo de este prototipo se evaluó el compilador JET de Excelsior, en su versión para Windows; el código compilado resultó ser aproximadamente el doble de rápido que el código Java original. Así, Java es lo suficientemente rápido en la ejecución del código; de todas formas, si en algún momento resulta demasiado lento, en la aplicación resultante construida se pueden reimplementar las partes vitales en

lenguajes nativos. Efectivamente, el código que necesita ser mejorado es el de resolución de la ecuación mediante elementos finitos, que ocupa un porcentaje muy alto de la ejecución del programa (del orden del 95 %). Este código puede ser implementado en C o Fortran y ser utilizado desde la aplicación Java utilizando JNI, que forma parte de la especificación estándar de Java. Así, se pueden reemplazar las partes que se necesiten acelerar por código en otros lenguajes, para aumentar la velocidad global de la aplicación. Sin embargo, en el momento de su construcción Java se presenta lo suficientemente rápido como para manejar la aplicación entera, por lo que se utiliza para la implementación de este prototipo y también del prototipo del módulo de toma de decisiones de redes de telefonía móvil.

6.3. Decisiones para redes de telefonía móvil

La implementación del prototipo de toma de decisiones para redes de telefonía móvil es similar a la del prototipo de toma de decisiones para redes de fibra óptica. Como éste, se basa en el prototipo de resolución de las ecuaciones en derivadas parciales explicado en la sección 6.1. Cuando se plantea el realizar una implementación del modelo de redes de telefonía móvil surgen una serie de decisiones que hay que tomar para poder construir un sistema válido:

- A la hora de resolver el problema de redes de telefonía móvil se plantea primeramente la opción de usar una malla de elementos finitos uniforme o no uniforme, como en el caso de las redes de fibra óptica. Dado que las ecuaciones involucradas son las mismas que en las redes de acceso a Internet el razonamiento a seguir es el mismo, de forma que a partir de la forma de las soluciones de las ecuaciones en derivadas parciales se deduce que las mejores mallas son las no uniformes que tienen mayor densidad en la parte izquierda del dominio del ancho de banda, debido a la mayor complejidad de la función en esa zona. La malla se construye con el método ya comentado en la sección 6.2.
- La principal decisión que se ha de tomar a la hora de implementar el modelo de redes inalámbricas se refiere al uso de una o varias mallas de elementos finitos, como en las redes de acceso a Internet. El uso de una sola malla presenta la ventaja de ser más sencillo, ya que solamente hay que calcular dicha malla una sola vez, y ese cálculo es válido para todos los clústers, de forma que no hay que realizar interpolación entre mallas; sin embargo, el uso de una malla tiene el inconveniente de tener que utilizar más puntos en el ancho de banda que el método con varias mallas para obtener la misma precisión, ya que cada una de las mallas individuales estará adaptada al ancho de banda de cada uno de los clústers, mientras que una malla única tendrá que cubrir todo el dominio de la capacidad de transmisión. Por otra parte, el uso de varias mallas presenta la ventaja de necesitar menos puntos, pero dos inconvenientes: la necesidad de interpolar una solución sobre las mallas de otros clústers y el problema de la influencia de actualizaciones fuera del ancho de banda máximo de

un clúster (ver sección 6.2). Por estos motivos, se utiliza una única malla para todos los clústers.

- Otra decisión que se ha de tener en cuenta es la del ancho de banda máximo para el cual se resuelven las ecuaciones del problema. Como se utiliza una sola malla, ésta tiene que englobar a todos los valores posibles de ancho de banda para todos los clústers, de forma que si se consideran n clústers ordenados de menor a mayor capacidad, el ancho de banda máximo a considerar tendrá que ser como mínimo igual a \bar{Q}_n . Además, el valor máximo se debe seleccionar lo más pequeño posible, para que la distancia entre puntos de la malla sea lo más pequeña posible y por tanto la precisión mayor. De forma empírica se comprueba que el mejor valor para Q_{max} es de 6 veces la capacidad máxima de transmisión del clúster n .
- Otro aspecto a analizar es la expresión concreta de la función de coste que se va a utilizar. La función de coste $C_{j \rightarrow u}(t)_{x \rightarrow y}$ es la que se encarga de calcular el coste de cada uno de los plazos en los que se divide el pago de los costes de mejora de un clúster, como ya se comentó en la descripción del modelo de redes de telefonía móvil. La función considerada en el prototipo es la que reparte en cantidades iguales los l plazos:

$$C_{j \rightarrow u}(t)_{x \rightarrow y} = \frac{C(t)}{l}$$

teniendo en cuenta que $C(t)$ es el coste de la mejora en el momento t . Este coste es el coste base de mejora (el coste en el momento actual, o $t = 0$) \bar{C} ajustado con el factor de decaimiento temporal.

Sin embargo, la expresión para $C_{j \rightarrow u}(t)_{x \rightarrow y}$ es incompleta, debido a que no considera la influencia de la tasa de interés libre de riesgo. El interés ha de ser considerado, de forma que el valor de C se ajusta mediante el factor

$$e^{-r (t_u - t)}$$

con t_u la fecha en la que la mejora que se está pagando se completa y t el instante de tiempo en el que se calcula el pago a realizar. Aplicando este factor corrector, se consigue que todos los plazos sean realmente de la misma cuantía, y la expresión completa de la función pasa a ser

$$C_{j \rightarrow u}(t)_{x \rightarrow y} = \frac{\bar{C} \times e^{-\alpha t}}{l} \times e^{-r (t_u - t)}$$

Estas decisiones son las que se han evaluado durante el proceso iterativo de desarrollo del prototipo para la toma de decisiones en redes de telefonía móvil. Debido a la forma de validación del prototipo, no se pudo probar de forma práctica la influencia de cada una de estas decisiones aisladamente, de forma que las diferentes iteraciones debían probar configuraciones completas de las decisiones: debido a las múltiples combinaciones posibles, el proceso de validación junto con la toma de las decisiones finales requirió una gran parte del proceso de desarrollo del prototipo.

Para la validación del software solamente se disponía del artículo original del que se extrajo el modelo de redes de telefonía móvil [5]. En este artículo se presentan una serie de resultados en forma de tablas, que se intentaron reproducir con el software construido. La validación se consideró completada cuando los resultados obtenidos con el prototipo fueron similares a los de las tablas del artículo, y cuando en dichos resultados se observaron los mismos efectos que en el artículo original.

6.4. Sistema de cálculo distribuido

Para realizar el cálculo distribuido se desarrolla una librería que permita distribuir el cálculo a través de una red de ordenadores. El principal requisito es la sencillez de uso, tanto a la hora de programar aplicaciones como a la hora de ejecutarlas en una red real. El requisito de sencillez surge como necesidad en un entorno en el que los usuarios no están especializados en informática, dado que un sistema complicado o incómodo conduciría inevitablemente a su progresivo abandono por parte de los usuarios.

Los sistemas de paralelización como PVM o MPI son muy potentes, permitiendo ejecutar tareas de forma paralela muy eficientemente, pero son complejos tanto a la hora de programar aplicaciones como de ejecutarlas. Sistemas como Globus tienen desventajas como requerir que los nodos de la red estén exclusivamente dedicados a la resolución de problemas. De las herramientas analizadas ninguna cumple el requisito de sencillez, así que se desarrolla un nuevo sistema para realizar cálculos distribuidos en programas Java.

6.4.1. Requisitos del sistema de cálculo distribuido

Los requisitos del sistema son los siguientes:

- Se desea construir un sistema para realizar cálculos distribuidos en programas Java. Estos cálculos deberán poder ser programados de forma sencilla, y la ejecución en paralelo debe ser de fácil invocación por personal no especializado.
- El sistema debe ser ejecutable en una red de propósito general, que comunique ordenadores de propósito general; no obligará a adquirir ningún hardware especial.
- El sistema deberá ser aplicable a problemas generales, no estará diseñado para un único problema. Las tareas que se realizarán de forma distribuida serán de grano grueso, pero no se impondrán restricciones sobre el tipo de problemas o el número de tareas que se resolverán.
- La eficiencia en las transmisiones de datos por la red **no** será prioritaria, dado que el propósito inicial es la aplicación del sistema a paralelización de tareas **de alto nivel**, que se supondrán de larga duración. Así, el realizar cálculos paralelos de grano muy fino no está entre los propósitos del sistema a desarrollar. Por esta razón, se realizarán las comunicaciones de la manera más eficiente

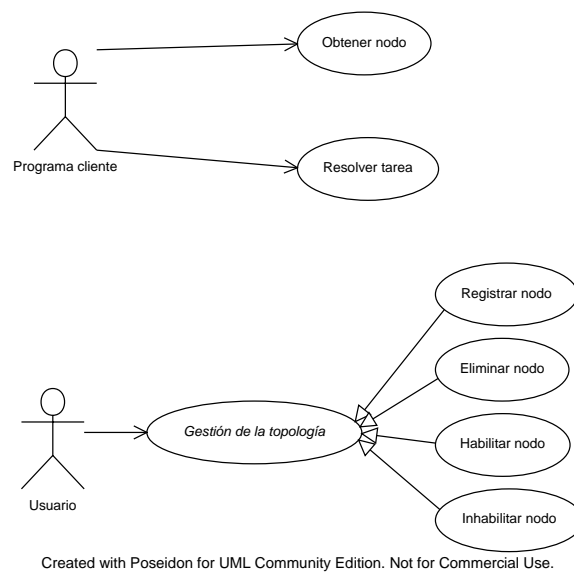


Figura 6.10: Casos de uso para el sistema de cálculo distribuido

que no aumente la complejidad en el uso del sistema. El hecho de no ser una prioridad la eficiencia de las comunicaciones hará que el rendimiento del sistema sea sustancialmente mejor cuanto más largas sean las tareas a resolver.

- Los ordenadores del clúster no deberán estar dedicados únicamente a la ejecución de las tareas programadas, sino que podrán ejecutar otros procesos de usuario simultáneamente.
- Los nodos del clúster se podrán añadir y eliminar del mismo de forma dinámica, sin tener que realizar tareas de configuración de ningún tipo.

6.4.2. Diseño del sistema

Los casos de uso identificados se muestran en la figura 6.10.

Se identifican dos actores, el programa cliente y el usuario externo.

- Programa cliente: se trata del programa Java que hace uso del sistema de cálculo distribuido para realizar algunas de sus tareas de forma remota.
- Usuario: el usuario representa un usuario humano, que configura el sistema para su funcionamiento.

Estos actores tienen acceso a tres grupos de casos de uso:

- Obtener nodo: este caso de uso representa la petición del programa cliente al sistema para reservar un nodo en el cual realizar cálculos posteriormente.
- Resolver problema: cuando el programa cliente ha reservado un nodo, puede requerirle para resolver una tarea.

- Gestión de la topología: es un caso de uso abstracto que engloba a todos los casos de uso que tienen que ver con la topología del clúster. Los requisitos especifican que estos casos de uso deben realizarse de forma dinámica, sin tener que reiniciar todo el sistema. Los subcasos son:
 - Añadir nodo: el usuario añade un nuevo nodo a la red, de forma que queda disponible para realizar cálculos remotos.
 - Eliminar nodo: retira un nodo de la red. Si se elimina el nodo central (el que administra todas las peticiones al sistema), la red debe reorganizarse de forma automática, sin ninguna intervención por parte del usuario.
 - Inhabilitar nodo: desactiva un nodo temporalmente, haciendo que no reciba más peticiones mientras esté desactivado.
 - Habilitar nodo: vuelve a activar un nodo que fue inhabilitado previamente, de forma que pueda recibir peticiones de nuevo.

Para ofrecer estos casos de uso el sistema se descompone en tres partes, la fachada cliente, el nodo central y el nodo de cálculo. Cada una de estas partes estará situada posiblemente en un ordenador de la red, aunque puede haber más de una parte en cada uno. En cualquier caso, solamente habrá un nodo central, que controlará a una serie de nodos de cálculo; en este nodo central los nodos de cálculo se registrarán y eliminarán según comiencen o terminen de ejecutarse. Además el nodo central estará encargado de conocer qué nodos están resolviendo qué tareas en un instante, y es el encargado de reservar y liberar dichos nodos de cálculo. Los nodos de cálculo, por su parte, expondrán una interfaz que permita resolver problemas genéricos, una vez reservado el nodo. Los programas cliente utilizarán cada uno una fachada cliente para acceder al sistema; esta fachada cliente les permitirá obtener nuevos nodos y ordenarles ejecutar tareas una vez obtenidos.

Método de comunicación

El método de comunicación entre las diferentes partes del sistema se eligió entre RMI o CORBA. Ambos sistemas son los principales sistemas de ejecución de métodos remotos más utilizados actualmente en programas Java, con diferentes características. Para el sistema en concreto se ha elegido CORBA, debido a que posee la característica de independencia de localización: los programas que acceden a objetos remotos no saben dónde están situados físicamente esos objetos en la red; solamente conocen la localización del servidor de nombres donde los objetos se registran. Esta característica permite que el sistema de paralelización sea mucho más flexible que si se tuviese que conocer la localización de cada uno de los nodos de la red; de esta forma, se puede reorganizar la red de forma transparente para los usuarios y programas clientes. Además de esta característica principal, otras funcionalidades que presenta

CORBA también pueden ser utilizadas en el sistema, como por ejemplo el servicio de transacciones o el de seguridad.

Nodo de cálculo

El nodo de cálculo es el elemento básico en el sistema de cálculo distribuido. Es el encargado de recibir peticiones de resolución remota de tareas, de ejecutar dichas tareas y devolver los resultados al origen de la petición. Para ello se definen interfaces IDL² comunes a los nodos que exponen las operaciones necesarias. Sin embargo, las operaciones que se pueden invocar en un nodo de forma remota se tienen que dividir en dos grupos: las operaciones que un cliente puede invocar en un nodo, y las operaciones que el nodo central puede realizar sobre el mismo nodo.

- Operaciones cliente: como ya se ha indicado antes, desde la parte cliente del sistema se va a invocar solamente una operación, que será la de resolver un problema. Esta operación requerirá dos parámetros, la especificación del problema (ver más adelante) y el identificador del cliente que realiza la petición (porque solamente podrá resolver una tarea en el nodo un cliente que lo haya reservado previamente). El método devolverá la solución al problema. Esta es la única operación que se invocará desde la fachada cliente sobre un nodo.
- Operaciones desde el nodo central: además de la operación de resolver problema, el nodo ha de presentar una serie de operaciones necesarias para el funcionamiento del sistema, a saber, una operación para reservar el nodo, una operación de ping y una operación de notificación de terminación. La primera de las operaciones es necesaria para que el nodo sea reservado por parte de un cliente y solamente ejecute la tarea que le indica dicho cliente; la operación de ping es necesaria para que el nodo central pueda saber cuándo un nodo de la red deja de ser accesible, y la última es necesaria para que el nodo central avise al nodo de cálculo cuando se desconecta (el nodo central; efectivamente, cuando el centro se desconecta, avisará a todos los nodos individuales para que la red se reestructure de forma automática). Sin embargo, estas operaciones no deben ser accesibles de forma pública dado que si lo fuesen un cliente malicioso podría reservar todos los nodos de la red de forma continua, sin dar lugar a un balanceo de las reservas de nodos entre todos los clientes, podría avisar a nodos de desconexiones del nodo central inexistentes (desestabilizando así la estructura del sistema) o incluso podría saturar la capacidad de comunicación de los nodos o incluso de la red inundando la misma de peticiones de ping. De esta forma, el acceso a estas operaciones debe estar restringido al nodo central, por motivos de seguridad.

Para resolver estos problemas, el nodo de cálculo separa su funcionalidad en dos fachadas, la fachada pública del nodo y la fachada privada. La fachada pública es a la que los clientes tienen acceso, ofreciendo

²IDL (*Interface Definition Language*) es el lenguaje en el que se especifican las interfaces de los objetos remotos que se van a comunicar por medio de CORBA

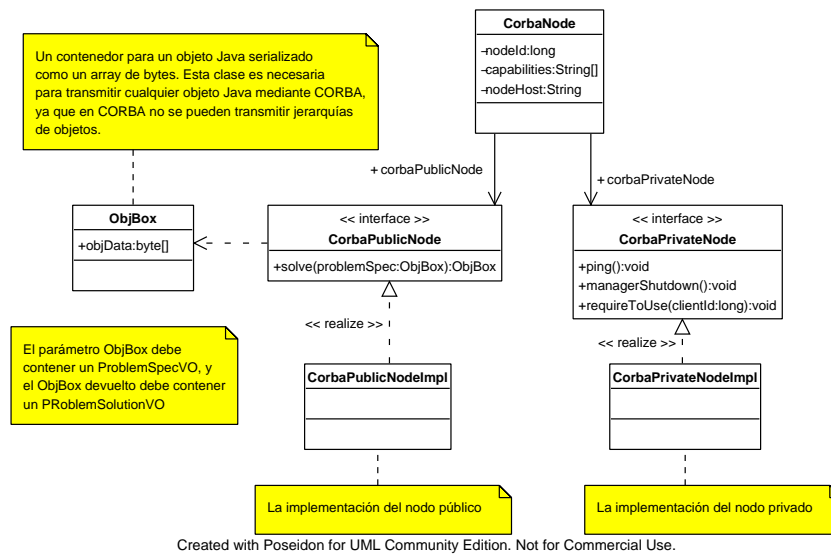


Figura 6.11: La especificación de un nodo de cálculo

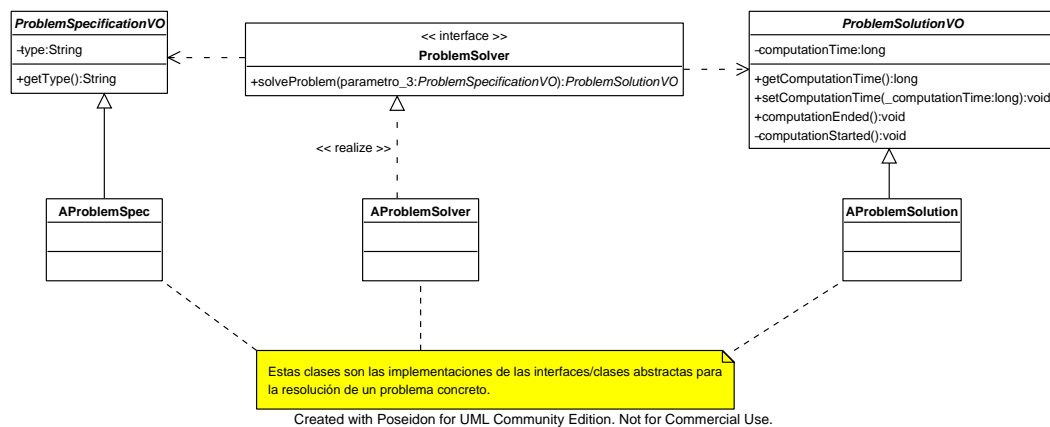


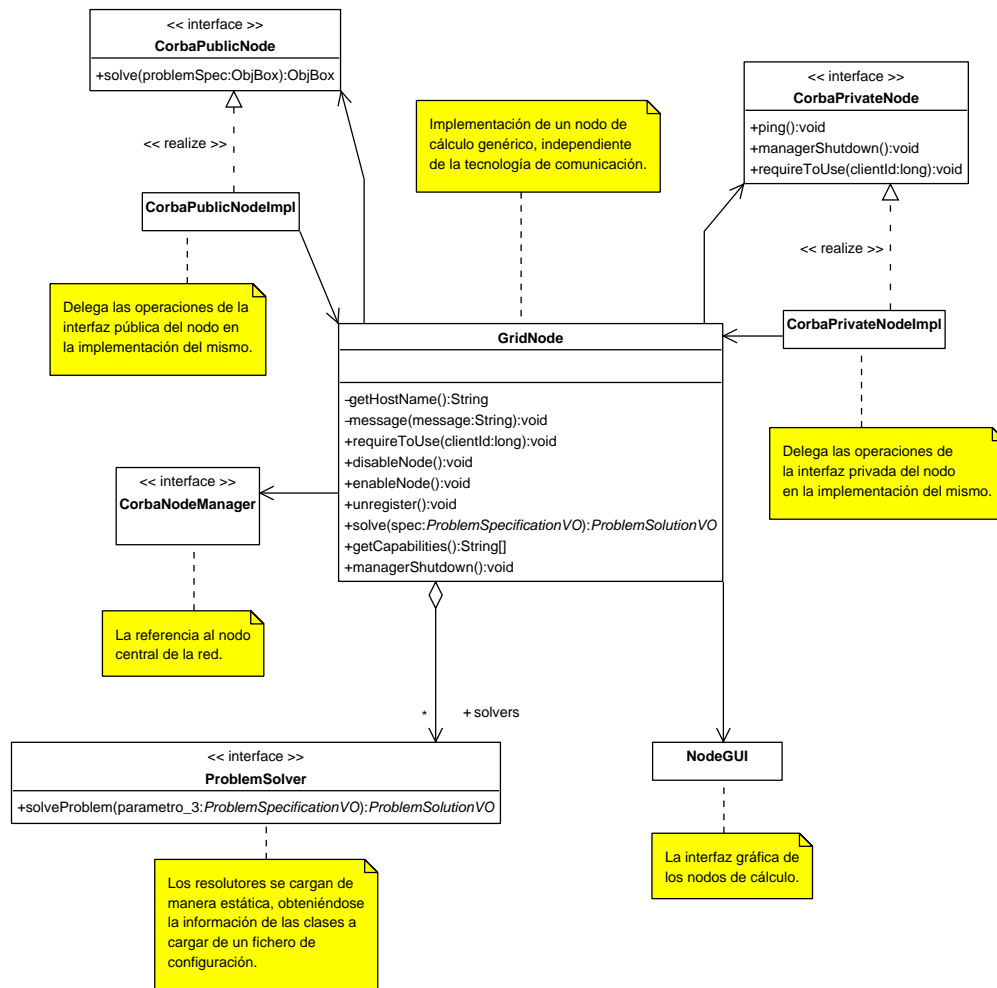
Figura 6.12: Clases para la especificación de un problema

solamente la operación de resolución de tarea; la fachada privada expone las operaciones que solamente debe ejecutar el nodo central. Se define también una estructura IDL para contener al nodo en su totalidad, y que se compone de las dos fachadas del nodo (pública y privada) y de información útil para el nodo central, como es el identificador único del nodo, el host en el que se está ejecutando o los tipos de problema que puede resolver (ver Figura 6.11).

Aparte de la infraestructura externa del nodo, que es la parte a la que los clientes y el nodo central pueden acceder de forma directa, el nodo de cálculo también define una estructura interna para la resolución de problemas. Efectivamente, dado que el sistema ha de ser genérico y aplicable a problemas no previstos, es obligatorio definir una interfaz común para las tareas a ejecutar, los resultados obtenidos, etc. Para ello se definen las interfaces y clases Java mostradas en la Figura 6.12:

- Especificación del problema (**ProblemSpecificationVO**): es una clase abstracta que define el tipo del problema. Todo problema que se desee resolver de forma distribuída tendrá que estar clasificado dentro de un tipo, y todas las especificaciones de ese problema estarán marcadas con el tipo correspondiente. Así, para cada tarea que se quiera resolver los clientes crearán un objeto de la subclase de **ProblemSpecificationVO** correspondiente al tipo de problema que se quiera resolver. Cada subclase será responsable de mantener los datos que sean necesarios para especificar completamente un problema a resolver.
- Solución del problema (**ProblemSolutionVO**): es una clase abstracta que indica solamente que un objeto es solución de un problema; cada problema a resolver tendrá como solución un objeto de una clase que subclasifique a **ProblemSolutionVO**. Además de los datos que cada una de estas subclases especifiquen, la clase **ProblemSolutionVO** contiene información acerca del tiempo de computación empleado en el cálculo de dicha solución.
- Resolutor de problemas (**ProblemSolver**): esta interfaz es la que define la operación de resolución de un problema; esta operación toma como parámetro un **ProblemSpecificationVO** y devuelve un **ProblemSolutionVO** después de realizar los cálculos pertinentes. Para cada tipo de problema que se quiera resolver, se creará una clase que implemente esta interfaz.

Estas clases permiten definir de forma genérica los conceptos de problema a resolver (tarea a ejecutar), especificación del problema y solución del mismo. Teniendo estas estructuras generales, el nodo realiza las tareas requeridas sin saber exactamente el tipo de problema que está resolviendo. Así, al arrancarse el nodo se le especifica mediante un fichero de configuración muy simple que problemas puede resolver ese nodo (las *capacidades* de ese nodo); estas capacidades serán una lista de tipos de problema, junto con los nombres de las clases que implementan o subclasifican las interfaces generales de los problemas (**ProblemSpecificationVO**, **ProblemSolutionVO** y **ProblemSolver**). Para cada uno de los problemas, el nodo crea el resolutor adecuado y lo añade a su lista interna de resolutores; cuando al nodo se le indica que resuelva un problema mediante una especificación, observa de qué tipo es la especificación. Si el nodo contiene un resolutor para ese tipo de problemas, entonces ejecuta el método del mismo que resuelve el problema, devolviendo el resultado obtenido. Si no conoce cómo resolver el problema, entonces avisará de este hecho mediante una excepción. Con esta configuración, es posible añadir de forma sencilla nuevos tipos de tareas ejecutables de forma distribuída, sin tener que recompilar ni modificar los nodos de cálculo; simplemente se desarrollarán las clases pertinentes para resolver dicho problema, y se le indicarán a los nodos de cálculo mediante su fichero de configuración, haciendo muy fácil el instalar nuevos tipos de problema (ver Figura 6.13).



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figura 6.13: Clases de la implementación de un nodo de cálculo

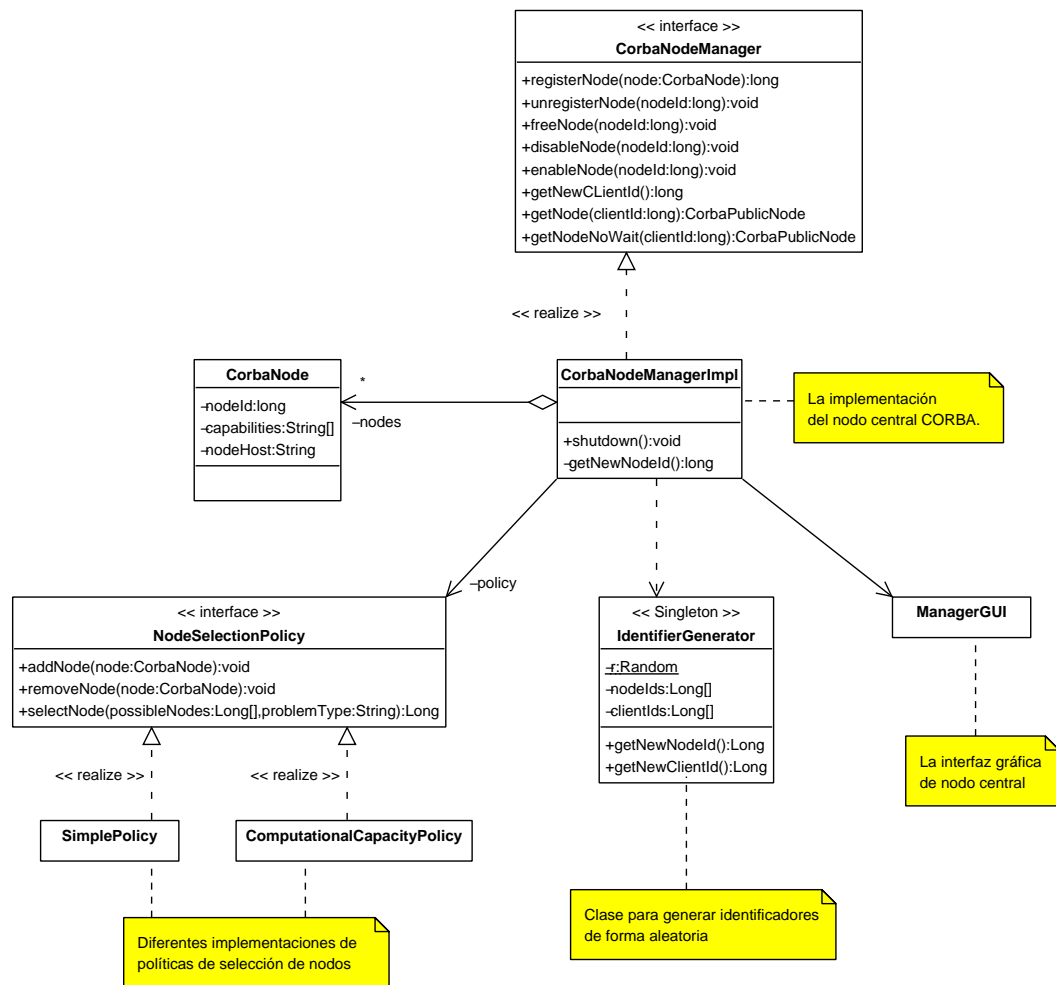
Nodo central

El nodo central del sistema es un nodo que no se utiliza para resolver tareas, sino que es el encargado de conocer la topología de la red, modificar dicha topología y administrar la concesión de nodos a clientes. Por ello es el punto por el que los clientes acceden al sistema.

Cuando el sistema arranca en un ordenador que está en una red que no contiene ningún nodo, detecta esta situación y arranca un nodo central, para que el resto de nodos puedan agregarse al sistema. Para ello, registra la interfaz del nodo central en el servicio de nombres CORBA, bajo un contexto fijo, para que el resto de nodos pueda encontrar esa interfaz y realizar peticiones en ella. Esta es la forma que tiene un nodo de saber si es el primero en añadirse al sistema (y por tanto debe comportarse como nodo central): un nodo se considera el primero si no encuentra un nodo central registrado en el servicio de nombres.

Los métodos expuestos por el nodo central están separados en dos grupos, los destinados a los nodos y los destinados a los clientes (verFigura 6.14):

- Los métodos destinados a los nodos son aquéllos utilizados para alterar la estructura de la red.
 - El método `registerNode` sirve para añadir un nodo al sistema. Este método recibe una estructura representando un nodo completo, con sus interfaces pública y privada, y le devuelve al nodo que la invocó un identificador único para ese nodo. Al ejecutar este método, el nodo central añade el nuevo nodo a la estructura de la red, tomando nota de las capacidades de ese nodo para la resolución de problemas, para poder saber qué nodos se le pueden proporcionar a un cliente para resolver un determinado problema.
 - El método `unregisterNode` es empleado por un nodo cuando termina su ejecución, para pedir al nodo central su eliminación del sistema. Para saber qué nodo ha de quitarse, el nodo le proporciona al sistema su identificador privado, que obtuvo mediante el método `registerNode`.
 - Cuando un nodo reservado por un cliente termina de resolver el problema que se le ha pedido, ha de notificar al nodo central que puede recibir peticiones de clientes de nuevo; para ello el nodo central dispone del método `freeNode`, que recibe el identificador del nodo que ha de ser liberado.
 - Los nodos de cálculo tienen la capacidad de desactivarse temporalmente, para poder reservar sus capacidades de cálculo para tareas ajenas al sistema de cálculo distribuido, sin tener que eliminar el nodo y tener que volver a añadirlo al terminar dichas tareas. Para notificar al nodo central la desactivación de un nodo, el nodo central dispone del método `diablNode`.
 - Cuando un nodo desactivado desea volver a estar disponible, debe notificarlo al nodo central; esto se hace mediante su método `enableNode`.



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figura 6.14: Especificación e implementación del nodo central

Estos métodos permiten modificar de forma dinámica la estructura del sistema, de forma que no ha de especificarse en ningún fichero de configuración una estructura fija. Esto permite añadir o eliminar de forma sencilla nodos de cálculo si se prevee una mayor o menor capacidad de cálculo, y también permite una mayor flexibilidad ante caídas de partes de la red.

- Por otra parte, los métodos destinados a los clientes representan las formas que tienen los clientes de acceder a la red.
 - Cuando un cliente desea acceder al sistema, debe primero obtener un identificador de cliente, para poder identificarse cuando desee ejecutar una tarea en el sistema. El método `getClientId` devuelve dicho identificador.
 - El principal propósito del nodo central desde la perspectiva del cliente es el de poder obtener referencias a nodos de cálculo; para ello el nodo central dispone del método `getNode`. Este método recibe un identificador de cliente, para que sólo el cliente que lo reservó pueda utilizar el nodo para resolver un problema, y el tipo de problema que el cliente desea resolver, para asegurar que el nodo obtenido por el cliente pueda resolver el problema deseado. El método devuelve un nodo de cálculo, pero solamente la interfaz pública del mismo, que es la que el cliente utilizará. Sin embargo, puede darse el caso de que en el sistema no haya ningún nodo que sepa resolver el problema en cuestión (debido a que ninguno de ellos posea el tipo del problema entre sus capacidades); en este caso, el método devolverá una excepción, concretamente una `CapabilityNotFoundException`.
 - Cuando un cliente pide un nodo para un problema del cual el sistema sabe que se puede resolver mediante algunos de los nodos que tiene registrados, puede ocurrir que ninguno de los nodos válidos esté disponible, bien por estar resolviendo tareas para otros nodos, bien por estar temporalmente desactivados. Si esta situación se produce, el cliente deberá esperar a que un nodo quede libre, y luego recibirá el nodo pedido. Este es el comportamiento del método `getNode`, que se bloquea hasta que haya un nodo disponible. Sin embargo, se puede dar el caso de que un programa cliente no quiera esperar a que haya un nodo libre, porque desea empezar a resolver el problema por sí mismo, porque desea informar al usuario, o por cualquier otra razón. Por ello el nodo central dispone de otro método para el cliente, `getNodeNoWait`, que tiene los mismos parámetros que el método `getNode`, y sirve para el mismo propósito, pero con la diferencia de que no es bloqueante; cuando el nodo central no dispone de ningún nodo válido en el momento de la llamada, el método `getNodeNoWait` devuelve `null`.

El nodo central puede aplicar una serie de políticas a la hora de establecer la estructura de la red, y de proporcionarle los nodos a los clientes. Efectivamente, se pueden tratar los nodos de cálculo de forma homogénea, sin hacer diferencias entre ellos. Con esta política se le otorgará a cada cliente el primer nodo

libre que esté libre para el problema que se quiera resolver. Sin embargo, el nodo central puede mostrar un comportamiento más inteligente, mediante la aplicación de políticas más especializadas. Para ello se define la interfaz `NodeSelectionPolicy`, que establece un método para seleccionar un nodo para cada petición de un cliente. Cada nueva clase que implemente esta interfaz representará una nueva forma de seleccionar los nodos bajo las peticiones de los clientes. Junto con la implementación de la política ya comentada (consistente en seleccionar el primer nodo libre, implementada en la clase `SimplePolicy`) se proporciona otra política más refinada, que selecciona los nodos mediante su *calidad*. La calidad de un nodo se define como una aproximación de su capacidad computacional, de forma que sea una medida válida para comparar diferentes nodos. La nueva política, `ComputationalCapacityPolicy`, devuelve siempre a los clientes el nodo válido disponible que posea una mayor calidad, obteniéndose una mejor resolución de problemas en redes heterogéneas con nodos de muy diferentes capacidades computacionales. Para calcular la capacidad de un nodo, el nodo central le ordena la resolución de un problema tipo y establece la calidad del nodo como la inversa del tiempo que el nodo emplea en la resolución de dicho problema tipo. El problema utilizado es la búsqueda de los números primos entre 0 y 100000; este problema se detalla en la sección dedicada al sistema de cálculo distribuido en el capítulo de resultados obtenidos (sección 7.4.1). Al estar definidas de forma genérica mediante una interfaz, se pueden desarrollar nuevas políticas e integrarlas de forma sencilla en el sistema, optimizando su funcionamiento de la forma deseada.

Con los métodos especificados el nodo central puede atender de forma satisfactoria las peticiones de los otros dos componentes del sistema, los clientes y los nodos de cálculo.

Fachada cliente

La última parte que interviene en el funcionamiento interno del sistema es la fachada cliente. Este subsistema es el que los programas cliente utilizan para acceder al sistema de forma genérica, sin conocer la estructura del mismo ni los métodos de comunicación y gestión de nodos. Así, la parte cliente proporciona unas clases que desacoplan los programas clientes del sistema de cálculo distribuido, permitiendo así cambiar en un futuro la tecnología de comunicación a RMI o a otra tecnología similar, en lugar de CORBA, o pudiendo remodelar todo el sistema para que utilice una arquitectura completamente diferente (por ejemplo para que internamente emplee un sistema de paralelización ya existente); estos cambios no afectarán al código de los programas que utilicen el sistema, gracias al sistema de fachada cliente.

La estructura que la fachada cliente expone a los programas cliente consta de dos clases principales, la clase `NodeManager` y la clase `Node`, junto con una serie de clases auxiliares, como las excepciones utilizadas (ver Figura 6.15).

La clase `NodeManager` es un envoltorio para la interfaz remota del nodo central. Esta clase tiene tres métodos, cada uno de los cuales se corresponde con uno de los métodos del nodo central destinados a los clientes (`getNewClientId`, `getNode` y `getNodeNoWait`). Sin embargo, los nodos que se devuelven

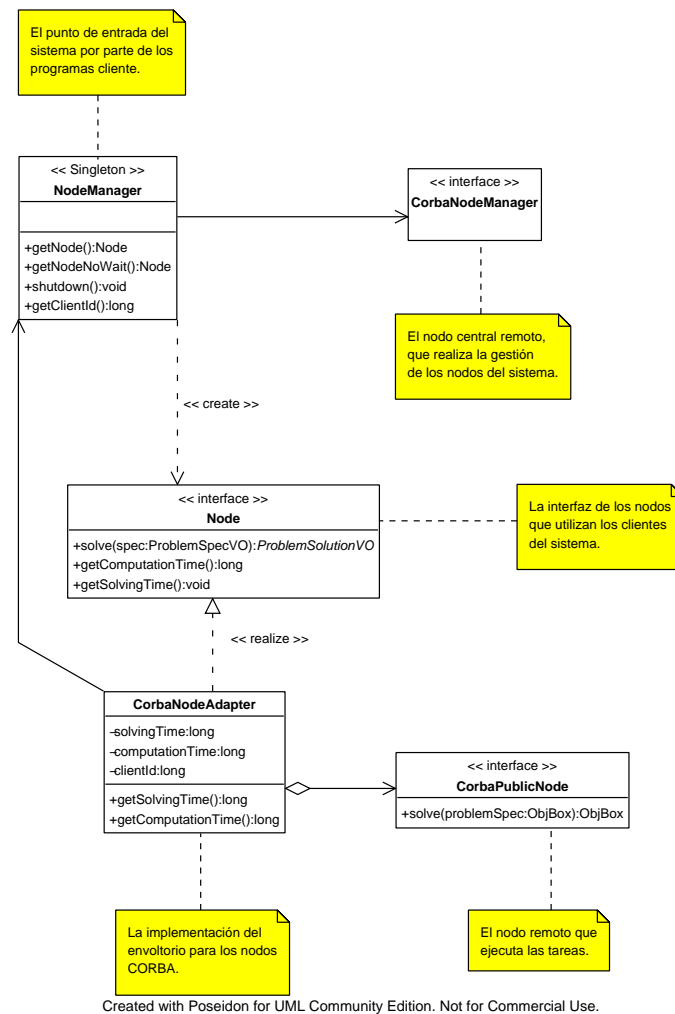


Figura 6.15: Clases de la fachada cliente del sistema

en los métodos de esta clase envoltorio no son las interfaces públicas remotas de los nodos de cálculo utilizadas en el sistema, sin que se envuelven mediante la clase **Node**. Esta clase proporciona el mismo método que el nodo que envuelve (`solveProblem`) pero con la diferencia de que no recibe el identificador de cliente, sino que se encarga de rellenarlo él mismo. Además, este envoltorio añade semántica al método de resolución de problemas. Efectivamente, el método `solve` de la interfaz **Node** lanza tres excepciones, en lugar de la única excepción que lanza la interfaz remota CORBA del nodo. La excepción que mantiene el significado de dicha interfaz remota es **UnknownProblemSpecException**, lanzada cuando al nodo se le pide que resuelva un problema de un tipo que no sabe cómo resolver³. Sin embargo, el envoltorio para el nodo lanza también la excepción **PermissionDeniedException**, que representa el hecho de que un cliente está intentando ejecutar una tarea remota en un nodo que no está reservado: la secuencia de hechos que lleva a utilizar un nodo no reservado es la siguiente:

1. El programa cliente pide un nodo al nodo central, y lo obtiene. Al devolvérselo, el nodo central avisa al nodo de cálculo de que está reservado por el cliente.
2. El programa cliente resuelve una tarea remota en el nodo reservado. Al finalizar la ejecución, el nodo remoto se libera automáticamente avisando al nodo central, pasando a estar en un estado no reservado.
3. (Opcional) Otro cliente con otro identificador reserva el nodo para sus propios propósitos.
4. El primer cliente vuelve a resolver un problema en el nodo de cálculo, del que ha guardado la referencia. En este momento, se haya reservado el nodo de cálculo por otro cliente o no (en todo caso, no está reservado por el cliente en cuestión), se lanza la excepción **PermissionDeniedException**.

Esta secuencia de eventos se muestra en el diagrama de secuencia de la Figura 6.16.

Esta excepción obliga a tener que reservar los nodos antes de utilizarlos. La otra excepción del método `solve` es la **InternalErrorException**, que se lanza cuando ha habido algún error del sistema (por ejemplo, un problema de comunicación con los objetos remotos) que no se puede sortear. Así, el sistema le da la oportunidad al programa cliente de tomar las acciones que considere necesarias, al no haberse completado la petición (por ejemplo, resolver el problema él mismo).

Mediante las clases **NodeManager** y **Node** el cliente puede acceder al sistema sin utilizar ninguna de las clases internas del sistema, ni las interfaces definidas en IDL ni las excepciones ni estructuras propias del sistema, ya que todos los tipos y excepciones internas se transforman en los tipos y excepciones definidos para los clientes. Esto permite, como ya se ha comentado anteriormente, modificar la implementación del sistema sin que los clientes que ya están creados se vean afectados, por lo que no necesitan ser modificados.

³esta excepción no debería ser lanzada nunca por este método, ya que significa que al sistema se le pidió un nodo para resolver un tipo de problema y al nodo se le pide que resuelva un problema de otro tipo; el nodo central solamente devuelve nodos cuyas capacidades incluyan el tipo de problema que el programa cliente quiere resolver.

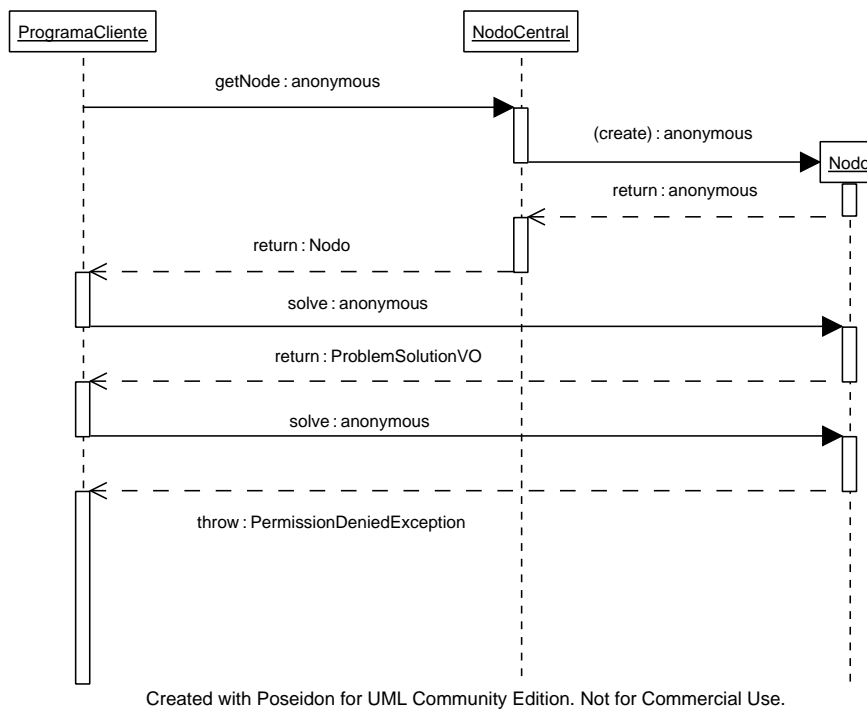


Figura 6.16: Una secuencia que produce una excepción `PermissionDeniedException`

Interacción entre las partes

Los tres subsistemas descritos (nodo central, nodo de cálculo y fachada cliente) colaboran para la correcta ejecución de los casos de uso definidos. Como se explicó anteriormente, dichos casos de uso están divididos en dos grupos, según sean invocados por un programa cliente o por un usuario del sistema. Los casos de uso invocados por un programa cliente involucran a las tres partes del sistema, mientras que los casos de uso invocados por los usuarios solamente afectan a los subsistemas de nodo central y nodo de cálculo.

Los casos de uso invocados por el usuario son los que establecen la estructura de la red, cuando dicho usuario ejecuta o termina el software del sistema en los ordenadores que se quieren emplear para la resolución distribuida de problemas.

- Inicio de la red: cuando se arranca un nodo, el software de ese ordenador comprueba la existencia de un nodo central registrado. Si no lo encuentra, considera que no existe y arranca un nuevo nodo central, registrándolo en el servicio de nombres para que el resto de nodos que se arranquen puedan encontrarlo.
- Adición de nuevos nodos: cuando un usuario arranca un nodo cuando hay un nodo central registrado, pasa a comportarse como un nodo de cálculo. Al hacerlo, obtiene la lista de problemas instalados en el nodo, y determina sus capacidades a partir de ellos. Una vez obtenidas, crea las interfaces

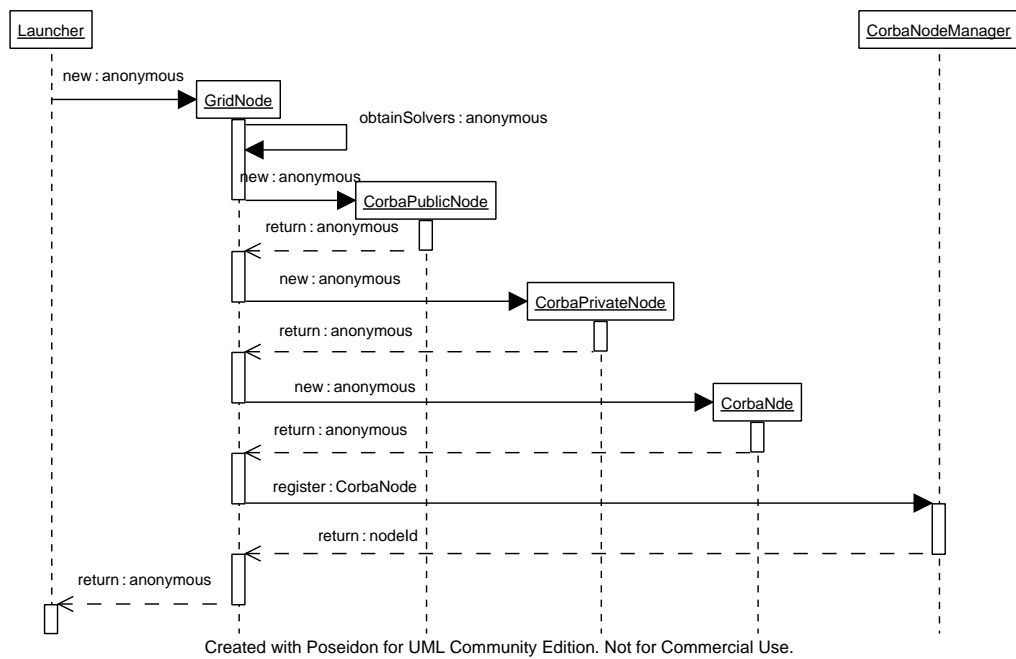


Figura 6.17: Secuencia de arranque de un nodo de cálculo

remotas necesarias y las registra en el nodo central, pasando a estar disponible para los clientes que lo requieran (Figura 6.17).

- Eliminación de un nodo: cuando el usuario lo desee, puede cerrar el programa que actúa como nodo de cálculo. Cuando este programa termina su ejecución por esa causa, antes realiza una llamada al nodo central para eliminarse del sistema de cálculo. De esa manera el nodo central sabe que no debe pasar más peticiones de clientes al nodo, ya que ha terminado su ejecución (figura 6.18).

Otra forma de que un nodo se elimine del sistema es que el ordenador en el que se ejecuta se caiga o quede separado de la red por cualquier causa ajena al sistema. En este caso, la situación no se

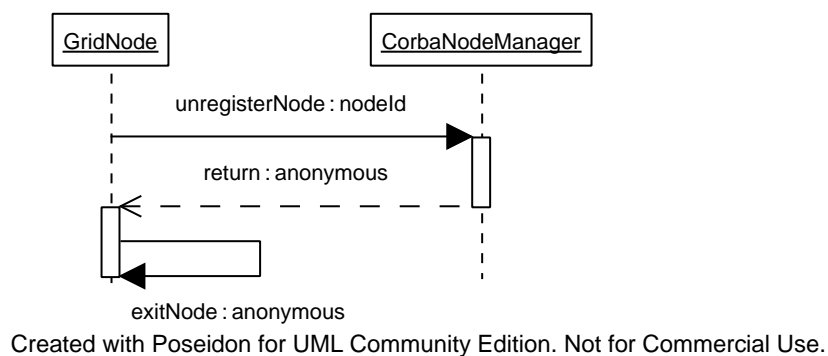


Figura 6.18: Finalización normal de un nodo

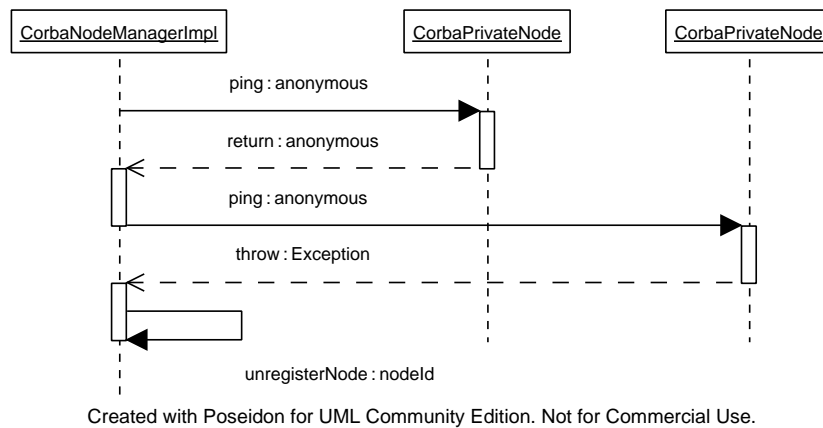


Figura 6.19: Eliminación de un nodo cuando deja de ser accesible

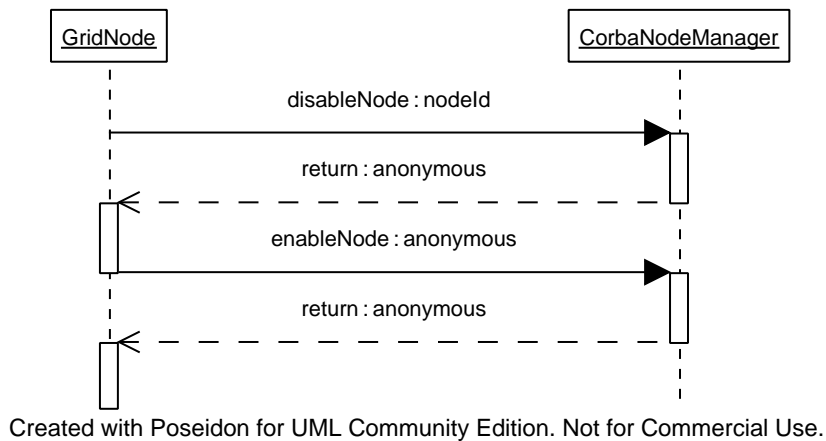


Figura 6.20: Proceso de desactivación y reactivación de un nodo.

detecta automáticamente, pero una de las tareas del nodo central es la de comprobar periódicamente la disponibilidad de los nodos que tiene registrados. Así, si un nodo no responde a las peticiones de ping del nodo central en un momento dado, es eliminado del sistema (ver Figura 6.19).

- Para la desactivación y posterior activación de los nodos de cálculo, el usuario selecciona la opción adecuada en el programa del nodo, y éste realiza la llamada pertinente al nodo central, a los métodos `disableNode` y `enableNode` (Figura 6.20).

Por otra parte, los casos de uso invocados por el programa cliente están orientados a la ejecución remota de tareas.

- La obtención de un nodo de cálculo es un caso de uso que involucra a las tres partes del sistema. En efecto, cuando un programa cliente necesita ejecutar una tarea remota, le pide un nodo al subsistema de fachada cliente. Éste le pide a su vez al nodo central que le proporcione un nodo para ejecutar

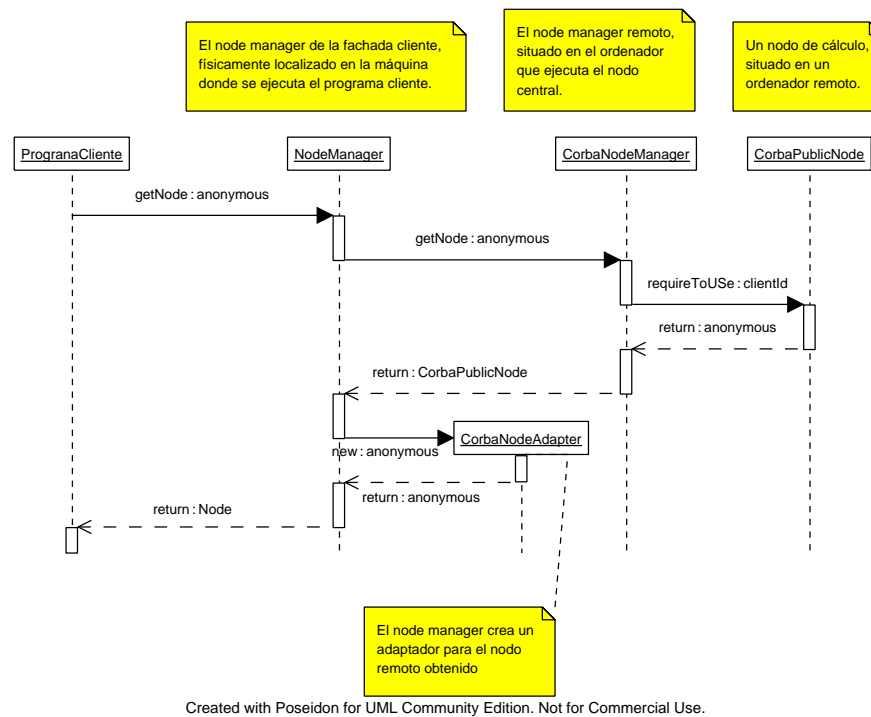


Figura 6.21: Obtención de un nodo por el programa cliente

la tarea, y cuando recibe la instancia remota del nodo, se la devuelve al programa cliente envuelta en la clase apropiada. Así, intervienen todas las partes del sistema colaborando para que el cliente obtenga el nodo pedido. Esta secuencia se muestra en la figura 6.21.

- El otro caso de uso invocado por el cliente es el de resolución de un problema de forma remota. En esta ocasión también colaboran los tres subsistemas para completar la ejecución del caso de uso, ya que el programa cliente le pide a la fachada cliente la ejecución de la tarea (mediante una llamada a la interfaz `Node`). La fachada cliente, por su parte, delega la ejecución de la tarea en el nodo de cálculo previamente reservado, pasando la ejecución a ese subsistema. Cuando acaba, y antes de devolver el resultado, llama al subsistema del nodo central para avisarle de que la ejecución terminó y el nodo está disponible para nuevas peticiones por parte de clientes. Así, los tres subsistemas realizan cada uno su parte en la correcta realización del caso de uso (Figura 6.22).

6.4.3. Utilización del sistema

La utilización del sistema de paralelización comprende dos grandes partes, la primera el establecimiento de la red de computación distribuida para que los ordenadores de la red puedan ser requeridos para cálculos remotos, y la segunda la programación y ejecución de aplicaciones que utilicen el sistema.

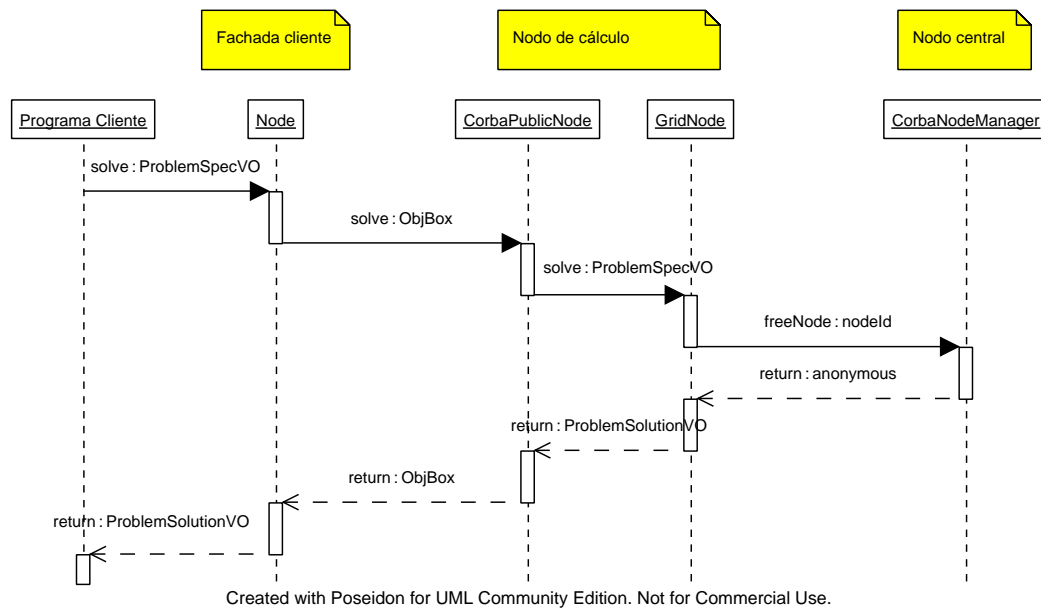


Figura 6.22: Secuencia para la resolución de un problema de forma remota

Establecimiento de la red

El establecimiento de la red de cálculo distribuido se realizará mediante las opciones que el programa del nodo ofrece al usuario para realizar los casos de uso que le conciernen. Antes de realizar cualquiera de estos pasos, es necesario que el servicio de nombres CORBA esté ejecutándose en alguno de los ordenadores de la red.

- Adición de un nodo: para añadir un nuevo nodo a la red, lo único que hace falta es ejecutar en un ordenador perteneciente a la red el programa del nodo. Este programa ha de configurarse inicialmente para indicar en qué ordenador está situado en servicio de nombres CORBA. Para ello, se le pasan como parámetros de ejecución el nombre de dicho servidor y el puerto en el que se está ejecutando el servicio de nombres, mediante los argumentos de línea de comandos -ORBInitialHost y -ORBInitialPort; por ejemplo, si el servicio está situado en el host `mipc` en el puerto 1030, los argumentos en la línea de comandos que hay que añadir son

-ORBInitialHost mipc -ORBInitialPort 1030

Una vez que se arranca el programa con estos argumentos, intentará comprobar si ya hay un nodo central. Si en el servicio de nombres no hay ninguno registrado, entonces arrancará un nuevo nodo central, lo registrará como el nuevo nodo central en el servicio de nombres y mostrará al usuario la interfaz de control del nodo central (ver Figura 6.23).

En ella se muestra un histórico de mensajes con todos los eventos que han sucedido en el nodo, y una lista de cada uno de los nodos que hay registrados en el sistema. Para cada uno de dichos

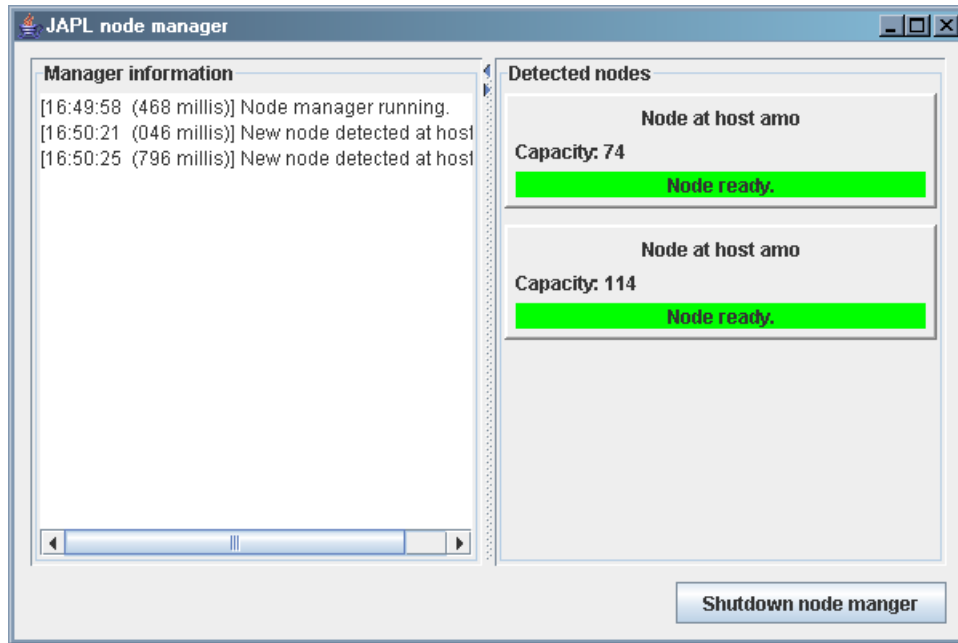


Figura 6.23: Interfaz gráfica para el nodo central

nodos la interfaz muestra su estado y una breve información. Además, se dispone de un botón para desconectar el nodo.

Sin embargo, si el nodo central ya existe cuando se arranca un nuevo nodo, entonces dicho nodo pasa a comportarse como un nodo de cálculo. En lugar de los pasos descritos anteriormente, el nodo arranca un nodo de cálculo y lo registra en el nodo central, y muestra al usuario la interfaz del nodo. En ella aparece un histórico con todos los eventos que han sucedido en el nodo, junto con estadísticas de utilización de dicho nodo. También se muestra una lista de los problemas *instalados* en el nodo, que representan sus capacidades. Además, se dispone de dos botones, uno para desactivar o reactivar el nodo y otro para desconectarlo (ver Figura 6.24).

De esta forma, para establecer la red de cálculo distribuido solamente hace falta ejecutar un programa en cada uno de los nodos que se desee agregar a la red, y ellos solos se configuran, mostrando unas interfaces gráficas que permiten su gestión de forma cómoda.

- Eliminación de un nodo de cálculo: para eliminar un nodo de cálculo de la red, el usuario solamente deberá pulsar el botón de desconexión de la interfaz gráfica de dicho nodo; automáticamente el nodo notificará al nodo central su desconexión y procederá a terminar su ejecución.
- Eliminación del nodo central (redistribución automática de la red): un caso especial de eliminación de un nodo se da cuando se termina el nodo central. Cuando esto sucede, el funcionamiento normal de la red debe detenerse, ya que todas las peticiones de reserva y liberación de nodos pasan por el nodo central; si este nodo no existe, el sistema no puede continuar. Por ello, al apagar el nodo

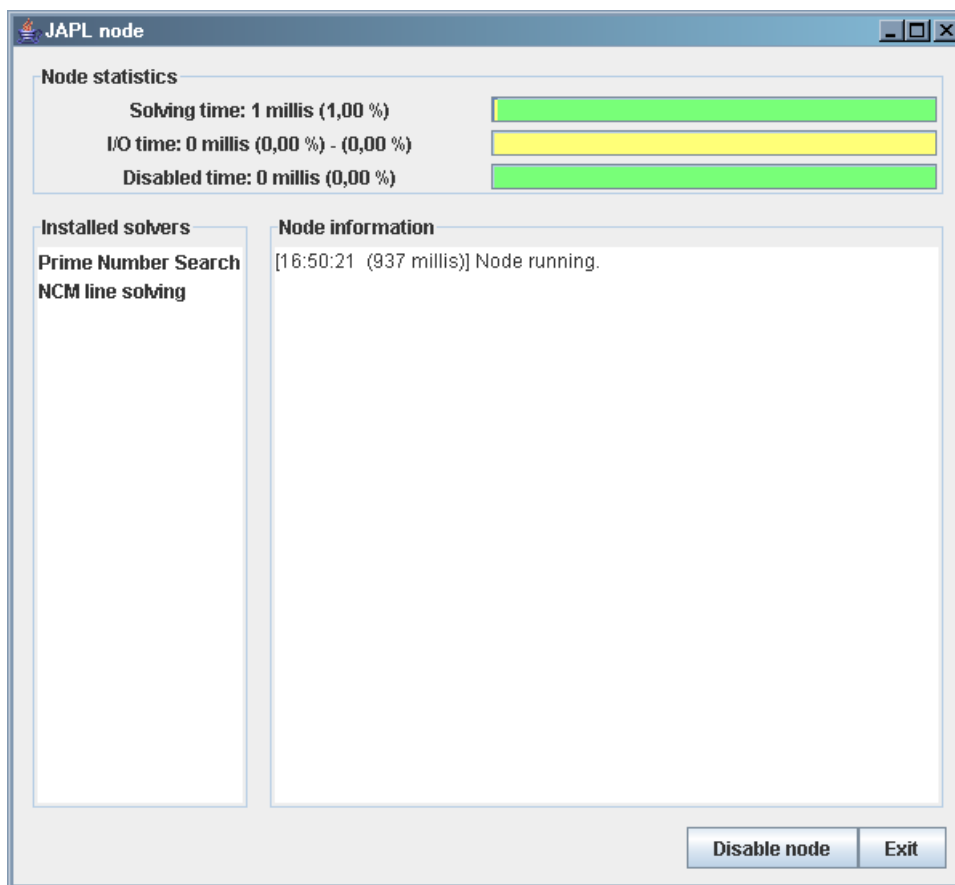


Figura 6.24: Interfaz gráfica para un nodo de cálculo

central el sistema ha de reaccionar de forma que la red se recupere de la mejor forma posible. Para conseguirlo, al apagar el nodo central éste avisa de ese hecho a todos los nodos registrados en el sistema. Cuando un nodo de cálculo recibe esta notificación, lo que hace es desconectarse y esperar un tiempo aleatorio; pasado ese tiempo, comprueba si ya se ha creado otro nodo central. En caso afirmativo, vuelve a registrarse en ese nodo central; en caso negativo, él mismo pasa a comportarse como nodo central. Esto provoca que al eliminar el nodo central de una red ya establecida, uno de los nodos de cálculo cambia su naturaleza para comportarse como el nuevo nodo central, y el resto de los nodos de cálculo vuelven a actuar como tales. Esto provoca que la red se reconfigure dinámicamente, sin tener que mediar intervención por parte del usuario.

- Desactivación de un nodo: la interfaz del nodo de cálculo dispone de un botón para desactivar dicho nodo. Al pulsarlo, el programa notifica al nodo central que ha sido desactivado y no recibirá más peticiones hasta que sea activado de nuevo. El nodo central mostrará en su interfaz la nueva situación del nodo.
- Reactivación de un nodo: para reactivar el nodo, el usuario solamente ha de pulsar de nuevo el botón de cambio de estado, y el nodo se reactivará tras avisar al nodo central, que actualizará la situación de dicho nodo en su interfaz.

Programación y ejecución de aplicaciones

La programación de aplicaciones que aprovechen las capacidades de cálculo distribuído del sistema desarrollado es una tarea simple, como se especificó en los requerimientos del sistema. El desarrollo de cada una de las aplicaciones que se quiera escribir se separa en tres pasos bien diferenciados:

1. Estudio del problema: en primer lugar, se han de identificar las subtareas del problema que se van a realizar de forma distribuida, identificando también cuáles son sus entradas (especificación del problema) y sus salidas (solución del problema). A partir de ahora se seguirán los pasos para un problema con una sola subtarea; para problemas con más de una subtarea a distribuir, hay que repetir los siguientes pasos para cada una de las subtareas. A la subtarea identificada se le asigna un nombre, que a partir de ahora se le llamara el tipo de la tarea. Siempre que se pida el tipo de la tarea, habrá que introducir este nombre, siempre de la misma forma exactamente (el tipo es sensible a mayúsculas y minúsculas).
2. Programación de la resolución de la subtarea: en este paso se programan las clases que definen la subtarea, su solución y la clase que resuelve dicha subtarea.

a) Se debe crear una clase que extienda la clase abstracta

`abm.solver.guiapp.model.spec.vo.ProblemSpecVO`, que representara la especificacion de la tarea. Esta clase contendra todos los datos de entrada de la tarea. La superclase requiere que en el constructor se le pase un `String` con el tipo de la subtarea.

- b) Tambien se debe crear una clase que extienda la clase abstracta

`abm.solver.guiapp.model.solution.vo.ProblemSolutionVO`, y que representa la solucion de la tarea a distribuir; se incluirán en esta clase todos los datos que conforman la solucion de la misma.

- c) Otra clase es necesaria para resolver la subtarea. Esta clase tiene que implementar

`abm.solver.guiapp.model.solver.ProblemSolver`, y tiene que tener un constructor vacio, para que el sistema la pueda instanciar de forma correcta. La subtarea se resolvera en el metodo `solve`, que toma como parametro un objeto de clase `ProblemSpecVO` (al que el solver tendrá que hacer un `downcast` a la clase concreta del problema) y devuelve un `ProblemSolutionVO` (que será un objeto de la subclase definida en el paso anterior). Si la especificación que se recibe como parámetro no es del tipo correcto, el método debe lanzar una `abm.solver.guiapp.model.exceptions.UnknownProblemSpecException`.

- d) Estas tres clases se deben empaquetar en un archivo JAR, que se debera incluir en el classpath de los nodos que se quieran arrancar.

- e) Antes de arrancar un nodo, se debe modificar el archivo `solvers.properties` que contiene su directorio raiz, añadiendo a la propiedad `node/availableSolvers` un nombre corto que se le asigne a la tarea (que no tiene por que ser su tipo). Este nombre se separa mediante una coma del resto. Ademas, es necesario añadir dos propiedades al archivo, para indicarle al nodo qué tipo es el de la tarea y cuál es la clase solver. En concreto, suponiendo que el nombre corto elegido sea `mitarea`, las propiedades que hay que añadir son `node/solver/mitarea/type` y `node/solver/mitarea/solverClass`. El valor de la primera debe ser el tipo de la tarea, y el valor de la segunda debe ser el nombre completo de la clase que implementa `ProblemSolver`. Una vez modificado el archivo, se podrá agregar el nodo al sistema (de la forma indicada anteriormente).

- f) Una vez realizados estos dos últimos pasos en los nodos que se quieran poner disponibles para la resolucion de la subtarea, se arrancan dichos nodos; en las interfaces gráficas de dichos nodos aparecerá en la lista de problemas instalados el nuevo problema desarrollado.

Una vez completados estos pasos, se puede ya pedir al sistema nodos que resuelvan la subtarea programada.

3. Programación de la aplicación: una vez completada la programación del código que resuelve las subtarear identificadas, se ha de programar la aplicación que resuelve el problema completo; esta

programación ha de incluir los siguientes puntos:

- a) En la clase que se encarga de resolver el problema completo hay que inicializar el sistema de cálculo distribuído antes de realizar ninguna operación remota. Para ello, basta una llamada al método `getInstance` de la clase `abm.japl.client.NodeManager`, que inicializa el sistema y proporciona una instancia del administrador de nodos.
- b) El cliente (la clase que se encarga de resolver el problema completo) debe obtener del administrador de nodos una identificación de cliente, llamando a su método `getNewClientId`. Esta identificación permitirá al cliente obtener nuevos nodos para resolver subtareas.
- c) Cuando el cliente necesite la resolución de una subtask, tendrá que obtener un nodo mediante uno de los métodos del administrador. Estos métodos son `getNode` y `getNodeNoWait`, como ya se ha mostrado anteriormente (recuérdese que el primero de ellos es bloqueante y el segundo no). Ambos métodos reciben como parámetros un `String` con el tipo de la subtask y la identificación del cliente, y devuelven un objeto `Node`, que permite resolver la subtask.
- d) Una vez que el cliente tenga un objeto de clase `Node`, le indicará cuál es la subtask a resolver mediante un objeto `ProblemSpecVO` que le pasará como parámetro al método `solve`. La solución de la tarea se obtendrá como un `ProblemSolutionVO`. Cada una de las llamadas para la obtención de un nodo y la posterior resolución de una tarea debería realizarse en un `thread` propio, para que se efectúe de forma paralela al resto del código.

Con estos sencillos pasos, las tareas identificadas como paralelizables podrán ser resueltas de forma distribuída con un mínimo esfuerzo de codificación. Al ejecutar la aplicación cliente, el sistema se encargará de proporcionar los nodos a la aplicación según le sean requeridos, y dichos nodos resolverán las tareas utilizando los recursos de los ordenadores en los que hayan sido arrancados.

El código que se incluye en el método `solve` será similar al que se encargaría de resolver la subtask en una aplicación que realizase sus cálculos en un solo ordenador. Sin embargo, han de tenerse en cuenta dos restricciones a la hora de programar este código.

- En primer lugar, la clase de resolución de la subtask no puede mantener estado. Esto quiere decir que no debe tener ningún atributo que se utilice desde el método `solve`, ya que cuando un cliente solicita un nodo al sistema, puede recibir cualquiera de los nodos válidos libres, y no está garantizado que recibirá un nodo en concreto. Por ello, el código de cálculo de la subtask no debería guardar estado que ataña al cliente que le ha pedido la ejecución de la tarea, ya el siguiente cliente que requiera la resolución de una tarea del mismo tipo podría no ser el mismo. Así, la clase solamente deberá guardar un estado que compartan *todos* los nodos.

- En segundo lugar, desde la clase que resuelve la tarea se puede llamar al sistema de cálculo distribuido, para resolver la subtarea en subtareas paralelas más pequeñas. Esto es perfectamente posible, y puede ser deseable en muchos casos, pero ha de tenerse en cuenta que para obtener un nodo desde el código de resolución de la subtarea se ha de utilizar el método `getNodeNoWait` en lugar de `getNode`. Esto es así debido a que si se llama al método bloqueante se pueden producir situaciones de interbloqueo, al esperarse por un nodo que nunca podrá estar disponible.

Esta situación se puede visualizar fácilmente en un entorno en el que sólo existe un nodo de cálculo que puede resolver dos tipos de problema. Para la resolución del primer tipo, el nodo necesita la resolución de varios problemas del segundo tipo. La aplicación principal pedirá en algún punto de su ejecución un nodo para resolver una subtarea del primer tipo, y recibirá una referencia al único nodo disponible. Cuando dicho nodo comience a resolver la tarea, necesitará uno o más nodos que resuelvan subtareas más pequeñas del segundo tipo. Si estos nodos se piden mediante el método `getNode`, el nodo esperará a que quede disponible, pero el único nodo de la red es él mismo. Esta es una clásica situación de interbloqueo, ya que el nodo no puede terminar la resolución de la tarea que se le encargó hasta que reciba referencias a un nuevo nodo, y esta referencia solamente se podrá obtener cuando el nodo termine la resolución en curso. Por ello, la obtención de otros nodos desde la resolución de tareas en un nodo de cálculo siempre deberá realizarse con el método `getNodeNoWait` en lugar de `getNode`.

6.5. Aplicación final

La aplicación final a realizar tiene como propósito el proporcionar una forma cómoda para los usuarios no especializados en informática y métodos numéricos de resolver los problemas estudiados. La introducción de datos, la resolución del problema y la presentación y manejo de los resultados se realizará a través de una interfaz gráfica de usuario, que resulta mucho más cómoda que las interfaces de línea de comandos.

6.5.1. Análisis de requisitos de la aplicación

El software desarrollado tendrá que cumplir los siguientes requisitos:

- Debe ser sencillo de utilizar, tanto para personal con formación en informática y métodos numéricos como para usuarios sin dicha formación.
- Debe presentar una interfaz gráfica agradable para los usuarios.
- Debe permitir introducir los datos de entrada de forma cómoda.
- Debe permitir guardar en disco los resultados obtenidos, y también cargar resultados guardados



Figura 6.25: Casos de uso de la aplicación

anteriormente. Los datos de entrada serán guardados junto con las soluciones calculadas, y también serán cargados.

- Debe permitir imprimir los resultados calculados y/o cargados.
- Debe prestar soporte a los dos tipos de problemas estudiados, valoración de redes de fibra óptica y valoración de redes de telefonía móvil.

6.5.2. Diseño de la aplicación

Los casos de uso identificados para la aplicación software se muestran en la Figura 6.25.

El único actor identificado es el usuario, que es la persona que se encarga de manejar la aplicación. Es el actor encargado de ejecutar todos los casos de uso.

Como se puede comprobar, dichos casos de uso se dividen en cinco grupos:

- Introducción de datos: un caso de uso abstracto que representa el padre de todos los casos de uso de introducción de datos de un problema. Existen dos subcasos, uno para la introducción de datos

del problema de valoración de redes de fibra óptica y el otro para el problema de la valoración de redes de telefonía móvil.

- **Carga de datos:** este caso de uso abstracto engloba a todos los casos de uso en los que datos de un problema previamente introducidos en la aplicación se cargan de nuevo, sin tener que introducirlos uno a uno. Como en el caso de uso anterior, dos subcasos extienden a éste, uno para el problema de las redes de fibra óptica y el otro para el problema de las redes de telefonía móvil.
- **Resolver problema:** el caso de uso abstracto que se puede considerar principal. Representa la resolución de un problema dado, una vez introducidos los datos. También es extendido por dos subcasos, uno para cada uno de los tipos de problema resueltos.
- **Guardar datos y resultados:** este caso de uso abarca aquellas acciones que conllevan el guardar los resultados de un problema junto con sus datos de entrada, o bien solamente los datos de entrada. Como subcasos aparecen uno para las redes de fibra óptica y el otro para las redes de telefonía móvil.
- **Imprimir resultados;** por último, el caso de uso abstracto que representa la impresión de resultados para su manejo cómodo como informes. De nuevo hay dos subcasos, uno para cada tipo de problema que la aplicación resuelve.

Dicha organización de casos de uso se puede ver de otra forma: la aplicación presenta una infraestructura para la resolución de problemas, y para añadir un nuevo problema a la interfaz se deben proporcionar clases (si se está hablando de una aplicación Java) para ofrecer cinco casos de uso de ese problema, para la introducción y carga de datos, resolución del problema y almacenamiento e impresión de resultados. Una aplicación construida desde esta perspectiva definirá una serie de interfaces y/o clases abstractas que permitirán añadir nuevos problemas de forma sencilla; esto permite la reutilización de las partes comunes de la aplicación, permitiendo su fácil extensibilidad; de esta forma, se crea una aplicación con una estructura modular, en la que cada módulo estará encargado de la resolución de un tipo de problema (aunque se pueden crear módulos que resuelvan varios tipos, en principio lo normal será encontrar módulos para problemas individuales).

Por ello, la aplicación a construir se separa en tres partes: la infraestructura común, el módulo para la valoración de redes de fibra óptica y el módulo para la valoración de redes de telefonía móvil.

Infraestructura común

La infraestructura común de la aplicación consiste en una serie de clases que permitan ejecutar la aplicación, terminar dicha ejecución, y seleccionar el tipo de problema de entre los existentes para la resolución de sus casos de uso.

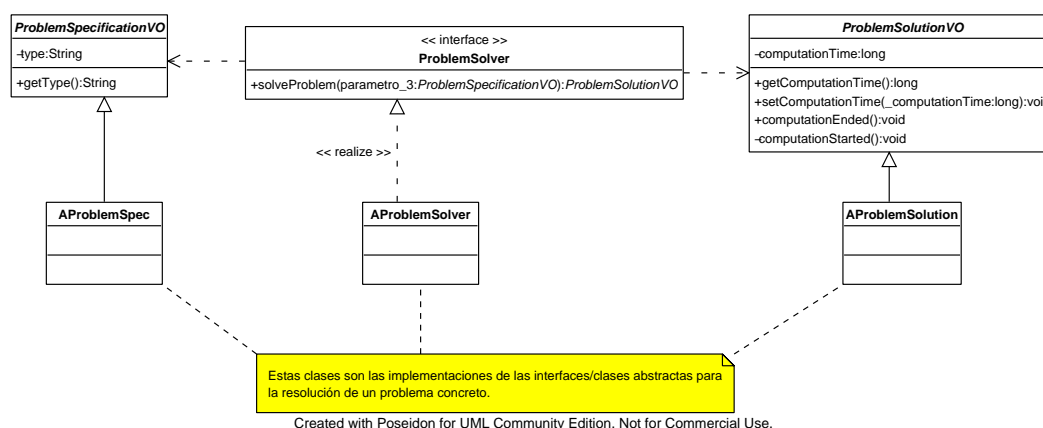


Figura 6.26: Clases para la especificación de un problema

En principio se define el conjunto de interfaces y clases abstractas que los nuevos módulos de resolución de problemas deben implementar o extender. La aplicación está orientada a la resolución de problemas, por lo que se decide reutilizar las clases definidas para la especificación de problemas, soluciones y resolutores en la sección del sistema de paralelización (6.4.2). Estas clases se muestran nuevamente en la Figura 6.26.

Como ya se explicó en la sección anterior, **ProblemSpecificationVO** contiene los datos que especifican un problema, **ProblemSolutionVO** contiene los datos que representan la solución a un problema y **ProblemSolver** es un resolutor que toma una especificación y devuelve la solución correspondiente. Estas tres clases son el núcleo de la aplicación, ya que se utilizarán en el caso de uso principal (el de resolución del problema). Además, la clase **ProblemSpecificationVO** presta soporte parcial a los casos de uso relacionados con los datos de un problema, ya que las diferentes subclases son las que mantendrán los datos de entrada de los diferentes problemas a resolver; la clase **ProblemSolutionVO** también se utilizará también en el caso de uso de guardar resultados, ya que sus subclases son las que contienen los datos de salida de los diferentes tipos de problemas resueltos.

Sin embargo, con estas clases no es suficiente para que los nuevos módulos puedan añadirse completamente a la aplicación. También se definen las clases mostradas en la Figura 6.27, que no eran necesarias para el sistema de paralelización:

- La introducción de datos es dependiente del problema a tratar, ya que claramente cada problema requiere unos datos de entrada distintos, así que se define una interfaz común para los componentes de entrada de datos, **ProblemInput**. Esta clase tendrá tres operaciones:

- **getJComponent**: con esta operación la interfaz gráfica tiene acceso al componente Swing⁴ que debe mostrar para que el usuario introduzca los datos del problema. Este componente

⁴Swing es una de las librerías estándar para creación de interfaces gráficas de usuario en lenguaje Java

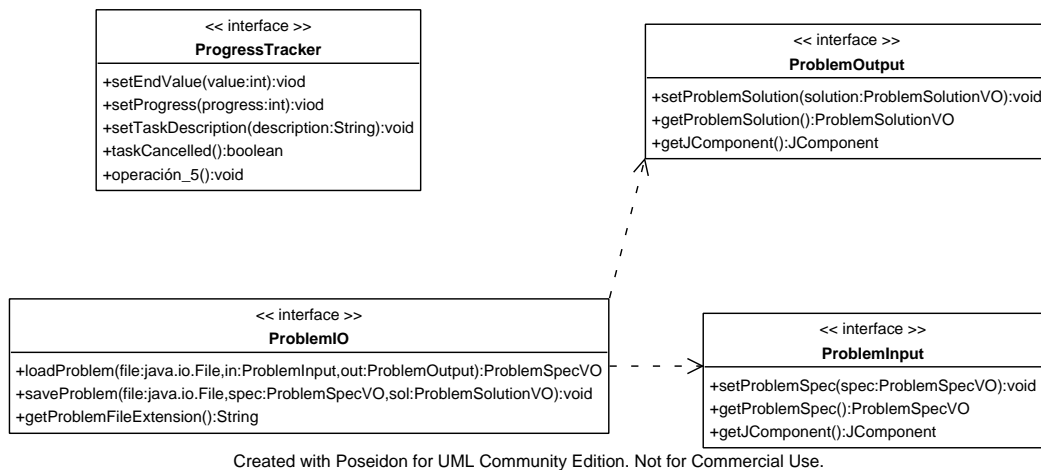


Figura 6.27: Clases adicionales para la definición de un módulo de resolución de un problema

vendrá definido por el problema concreto.

- **setProblemSpec**: con esta operación la aplicación establece la especificación del problema para el componente de entrada, lo cual tiene dos efectos: el componente de entrada de datos debe mostrar los valores de la especificación que se ha establecido, y dicho componente debe actualizar la especificación cuando el usuario modifique los valores de entrada en el componente gráfico.
 - **getProblemSpec**: con este método la aplicación obtiene una referencia al objeto de especificación que tiene asociado el componente de entrada.
- De la misma forma que la introducción de datos, el proceso de mostrar los resultados al usuario también es dependiente del problema en cuestión, por lo que se crea otra interfaz, **ProblemOutput**, encargada de la tarea de mostrar los resultados al usuario. Como en el caso de la clase encargada de la introducción de datos, esta interfaz tiene tres métodos definidos:
 - **getJComponent**: con esta operación la interfaz gráfica tiene acceso al componente Swing que muestra los resultados al usuario (este componente será dependiente del tipo de problema, como es normal).
 - **setProblemSolution**: sirve para que la aplicación especifique qué solución debe mostrar el componente.
 - **getProblemSpec**: mediante esta operación la aplicación tiene acceso al objeto solución que está mostrando el componente visual.
 - Para los casos de uso de carga y almacenamiento de datos, se define la interfaz **ProblemIO**, con tres métodos definidos.

- `getProblemFileExtension`: devuelve la extensión que deben tener los ficheros que almacenan datos de entrada o resultados del problema en cuestión.
 - `loadProblem`: este método recibe un fichero que contiene unos datos de entrada (y opcionalmente unos resultados) y carga dichos objetos (en un `ProblemSpecV0` y un `ProblemSolutionV0`). Una vez que los datos de entrada (y salida si los hay) han sido cargados, la especificación del problema se coloca en el objeto `ProblemInput` apropiado, y la solución (si se cargó) en el objeto `ProblemOutput` adecuado; de esta manera, se cargan los datos y se muestran al usuario.
 - `saveProblem`: a este método se le indica un fichero y unos datos a guardar. Obligatoriamente se ha de incluir una especificación de problema (mediante el parámetro `ProblemSpecV0`) y opcionalmente una solución a dicha especificación (mediante el `ProblemSolutionV0` adecuado). El método guardará los datos en el fichero.
- Por último, se define una interfaz para los indicadores de progreso. Un indicador de progreso es un componente que informa al usuario de cuál es el porcentaje actualmente completado de la tarea en curso; este tipo de componentes hace una interfaz mucho más amigable para las personas que la utilizan, ya que representan una indicación de cuánto va a durar la tarea que se está ejecutando, y dan la impresión de que la aplicación está resolviendo el problema: cuando se ejecutan tareas largas sin un indicador de progreso los usuarios no saben si la aplicación está funcionando correctamente o si por el contrario se ha producido un fallo y no se está llevando a cabo ningún cálculo, y de estarse ejecutando, los usuarios no saben cuánto durará la tarea. Por ello, se define la interfaz `ProgressTracker` que define operaciones para establecer el progreso máximo que puede tener una tarea, el progreso y descripción actuales de la tarea que se está ejecutando y también comprobar si la tarea ha sido cancelada por el usuario. El indicador de progreso será creado por la interfaz de la aplicación y se le dará una referencia al mismo al resolutor del problema, para que vaya indicando su progreso al usuario.

Con esta serie de clases se establece todo el conjunto de puntos que las clases que resuelvan los problemas a incluir en la interfaz deben implementar/extender. Como en el caso de los nodos de cálculo del sistema de cálculo distribuido, la aplicación sabrá qué problemas están instalados en la aplicación mediante la lectura de un fichero de configuración, que contendrá la lista de problemas que la aplicación puede resolver. Estos problemas estarán indexados por su tipo, que será un `String` determinando qué problema resuelve cada módulo. La clase que gestiona los problemas instalados es `ProblemCenter`, mostrada en la Figura 6.28. Esta clase obtiene de forma estática las clases de los problemas instalados, y las almacena indexándolas por su tipo. Cuando la aplicación necesita un nuevo objeto de especificación (`ProblemSpecV0`), de solución (`ProblemSolutionV0`), de resolución (`ProblemSolver`), de entrada (`ProblemInput`), de salida (`ProblemOutput`) o de carga y almacenamiento (`ProblemIO`) para un tipo de problema, se los pide a esta

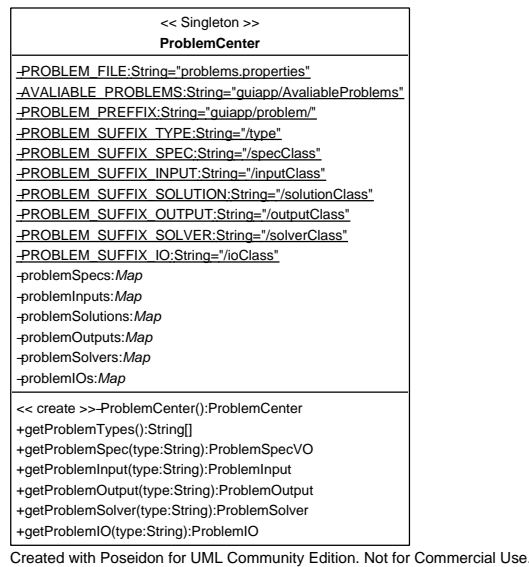


Figura 6.28: Clase que administra los problemas instalados en la aplicación

clase, que crea dichos objetos dependiendo del tipo de problema para el que sean requeridos. Además, el **ProblemCenter** también permite conocer todos los tipos de problemas instalados, mediante el método `getProblemTypes`, que devuelve la lista de tipos presentes.

Además de estas clases, que definen los puntos que los nuevos módulos deben extender, la aplicación debe proporcionar un marco donde los módulos añadidos puedan ejecutarse debidamente. Para ello, se construye una interfaz gráfica que interactuará con dichos módulos. Esta interfaz gráfica se compone de varias partes, definidas en la Figura 6.29, a saber:

- **SolverGUI**: la principal clase de la interfaz. Es una ventana MDI (*Multi Document Interface*), es decir, una ventana donde se pueden abrir múltiples subventanas. Cada nuevo problema a resolver posea una de estas ventanas, con lo que el usuario podrá manejar varios problemas al mismo tiempo. El aspecto inicial de esta ventana es el mostrado en la figura 6.30. Esta ventana tiene un menú desde el que se crean nuevos problemas. Al seleccionar la opción 'New problem', se crea y se muestra una instancia del **NewProblemDialog**.
- **NewProblemDialog**: este es el cuadro de diálogo que se muestra al usuario para seleccionar el tipo de problema que quiere manejar. Para ello, consulta al **ProblemCenter** para conocer qué módulos están instalados en la aplicación, y muestra una lista desplegable para que el usuario pueda elegir el tipo de problema (esta secuencia se muestra en la Figura 6.31; el cuadro de diálogo se puede observar en la Figura 6.32).

Una vez que el usuario acepta la selección, informa a la interfaz **SolverGUI** de cuál es el tipo de problema elegido, y éste crea el nuevo problema, mostrándolo a través de un **ProblemFrame**.

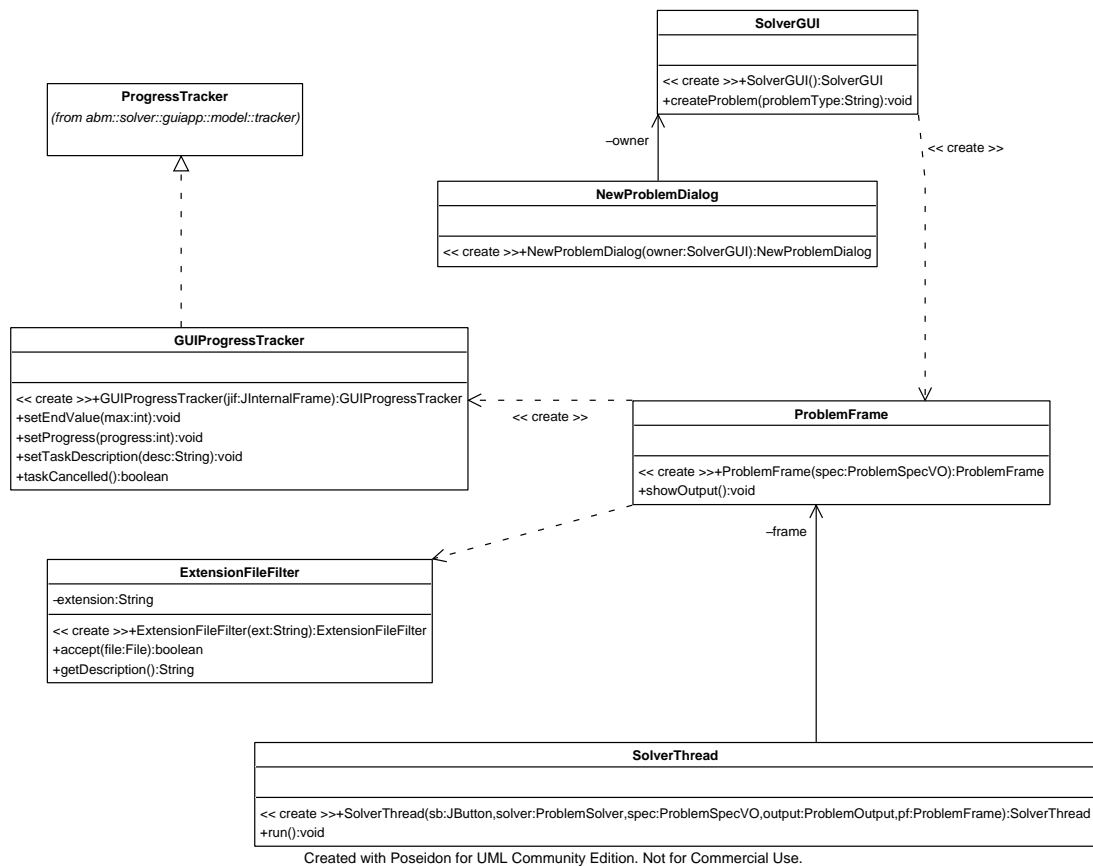


Figura 6.29: Las clases que definen la interfaz gráfica común

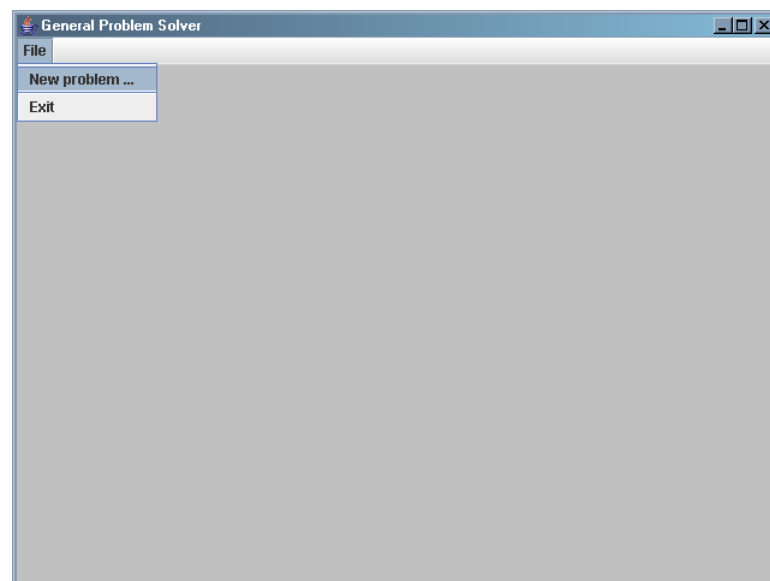


Figura 6.30: La interfaz gráfica de la aplicación, sin ningún problema abierto.

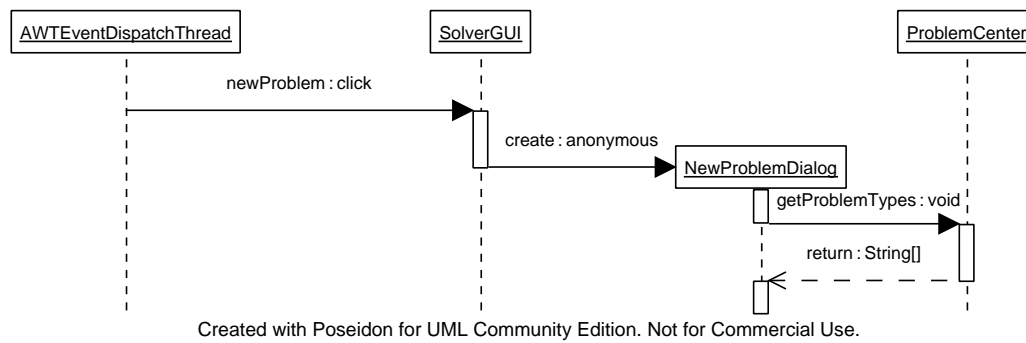


Figura 6.31: La secuencia de creación del cuadro de diálogo para un nuevo problema

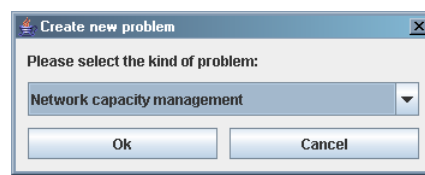


Figura 6.32: El cuadro de diálogo para crear un problema nuevo

- ProblemFrame:** es la ventana que permite al usuario introducir los datos de entrada para un problema, resolverlo y mostrar los resultados, además de guardar, cargar e imprimir datos (figura 6.33). Para crear una de estas ventanas, es necesario tener una instancia de la especificación del problema; a partir de este objeto **ProblemSpecVO** se crea el **ProblemFrame**. Para conseguir este objeto de especificación la ventana principal de la aplicación le indica al **ProblemCenter** cuál es el tipo del problema seleccionado, para que busque en el mapa de problemas instalados cuál es la clase concreta que tiene que instanciar.

Una vez que se conoce el tipo del problema, y se tiene la instancia de la especificación (que inicialmente tendrá valores por defecto), la ventana del problema tiene que construir su contenido. Para ello, conociendo el tipo del problema a través de la especificación que se le pasa, consulta al **ProblemCenter** para conseguir los componentes de entrada y salida (**ProblemInput** y **ProblemOutput**) para ese tipo de problema. Una vez que los tiene, los coloca en páginas distintas de la ventana del problema, para que el usuario pueda introducir los datos de entrada, y consultar los resultados (esta secuencia se puede observar en la Figura 6.34).

Además de los componentes de entrada y salida del problema, el **ProblemFrame** también dispone de un botón para que el usuario indique cuándo el sistema tiene que comenzar a resolver el problema. Para llevar a cabo la resolución, se crea una instancia de **SolverThread**, que es el hilo que ejecutará el trabajo.

Por último, cada ventana de problema tiene tres opciones más, incluidas en el menú “File”, que

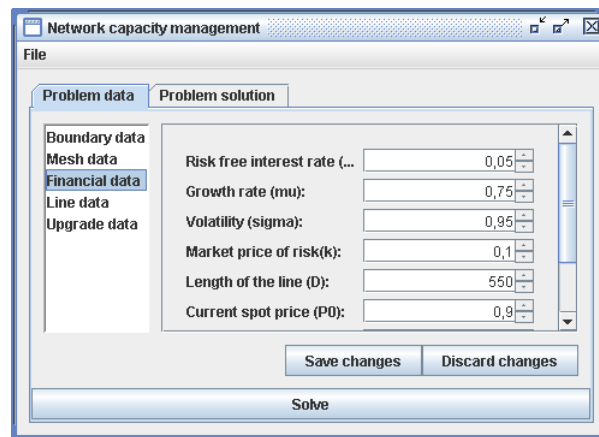


Figura 6.33: Una instancia de ProblemFrame

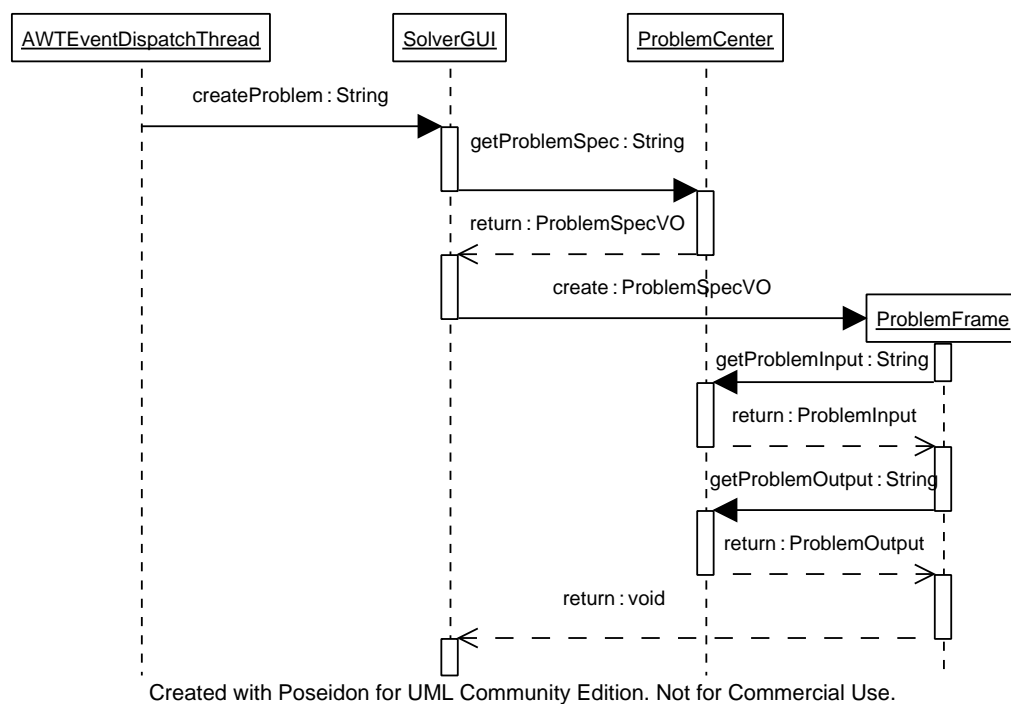


Figura 6.34: La secuencia de creación de una ventana de problema

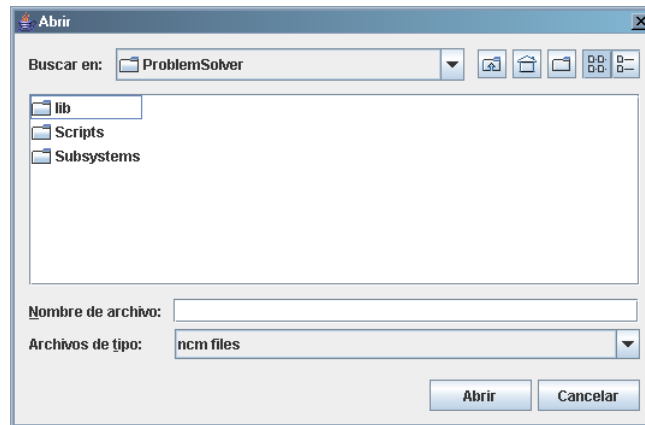


Figura 6.35: El cuadro de diálogo para selección de fichero

permiten cargar, guardar e imprimir los datos de la ventana. Al cargar unos datos, se mostrarán los datos de entrada y los resultados si fueron guardados también. Al guardar los datos, se guardarán los datos de entrada y los resultados si han sido calculados. Si el usuario decide imprimir los datos, se imprimirán todos los datos presentes en la ventana del problema. De estas operaciones, las de cargar y guardar datos tienen en común el hecho de trabajar con ficheros; estos ficheros deben ser seleccionados por el usuario, bien sea para crearlos o sobreescribirlos cuando se guardan datos, o para leerlos cuando se cargan. La infraestructura común de la aplicación incorpora la lógica de selección de ficheros: cuando el usuario pulsa en las opciones de carga o almacenamiento, el **ProblemFrame** se encarga de mostrar un cuadro de diálogo para que el usuario seleccione el fichero que desee (Figura 6.35); para ello, la ventana del problema consulta la extensión de los ficheros del problema (esta extensión la obtiene de la clase **ProblemIO**) y a partir de ella crea una instancia de la clase **.ExtensionFileFilter**, que usa para filtrar los archivos del sistema de ficheros, mostrando al usuario solamente los que son relevantes para el tipo de problema con el que está trabajando.

Una imagen de la aplicación con dos problemas abiertos se muestra en la Figura 6.36.

- **SolverThread**: este es el *thread* que resuelve el problema. Cada una de las ventanas de problema creará un hilo distinto para resolver cada problema; la resolución no puede llevarse a cabo en el hilo de manejo de eventos, debido a que si se dedica a eso no puede atender peticiones de la interfaz gráfica, por lo que ésta quedará bloqueada hasta que termine la resolución del problema, por lo que el usuario no podrá trabajar con varios problemas a la vez. Sin embargo, si se utiliza un *thread* separado para cada problema, el usuario puede comenzar la resolución de un problema y mientras no termina puede crear otros problemas y trabajar con ellos.

Para llevar a cabo la resolución del problema, el *thread* consulta al **ProblemCenter** para obtener una instancia del **ProblemSolver** apropiado para la clase de problema que está abierto. Cuando

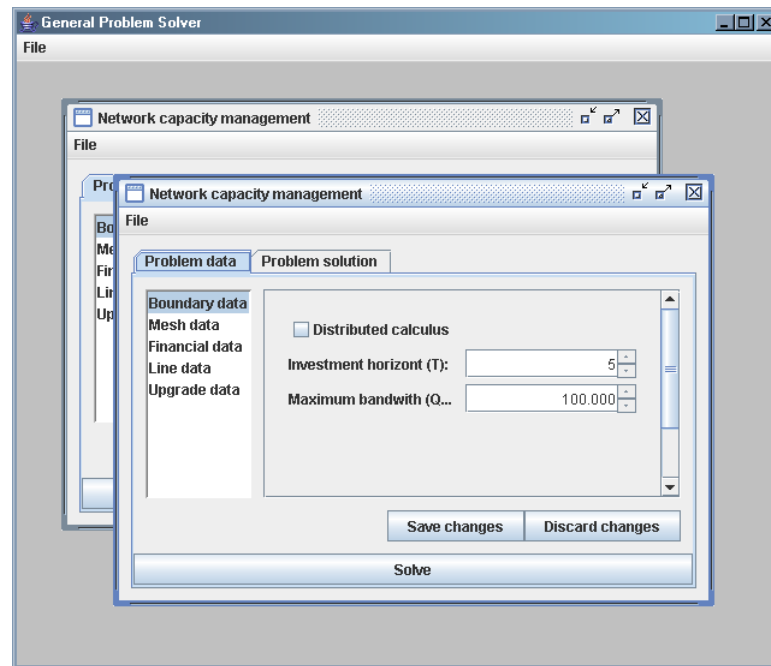


Figura 6.36: La aplicación con dos problemas abiertos

ese objeto está disponible, el hilo de resolución del problema le indica que comience la resolución del problema especificado por el objeto `ProblemSpecVO` que almacena el `ProblemFrame`; al finalizar la resolución, el resultado obtenido (un `ProblemSolutionVO`) se coloca en el `ProblemOutput` de la ventana correspondiente, para que pueda ser manejado por el usuario (ver Figura 6.37).

- **GUIProgressTracker**: esta clase es la implementación del indicador de progreso para la interfaz gráfica (Figura 6.38). Esta implementación presenta una barra de progreso al usuario, que le indica el porcentaje de la tarea que ha sido completado, además de una descripción de la tarea que se está ejecutando en cada momento. También se dispone de un botón para que el usuario cancele la tarea actual; esta opción solamente será efectiva si el método de resolución del problema hace uso de ella, a través del método `taskCancelled` de la interfaz **ProgressTracker**.
- **ExtensionFileFilter**: por último, la interfaz también hace uso de esta clase, que es un filtro para que la interfaz gráfica muestre al usuario solamente los archivos que tienen la extensión del tipo de problema que se quiere cargar o guardar, en el cuadro de diálogo de selección de archivo (Figura 6.35).

Finalmente, la última parte de la infraestructura común que queda por especificar es la encargada de comenzar la ejecución de la aplicación: para el arranque de la misma se define la clase `StartApp`, que simplemente crea una instancia de la interfaz gráfica `SolverGUI` y la muestra.

Una vez completado el desarrollo de la aplicación, se procede a la creación de los módulos para la

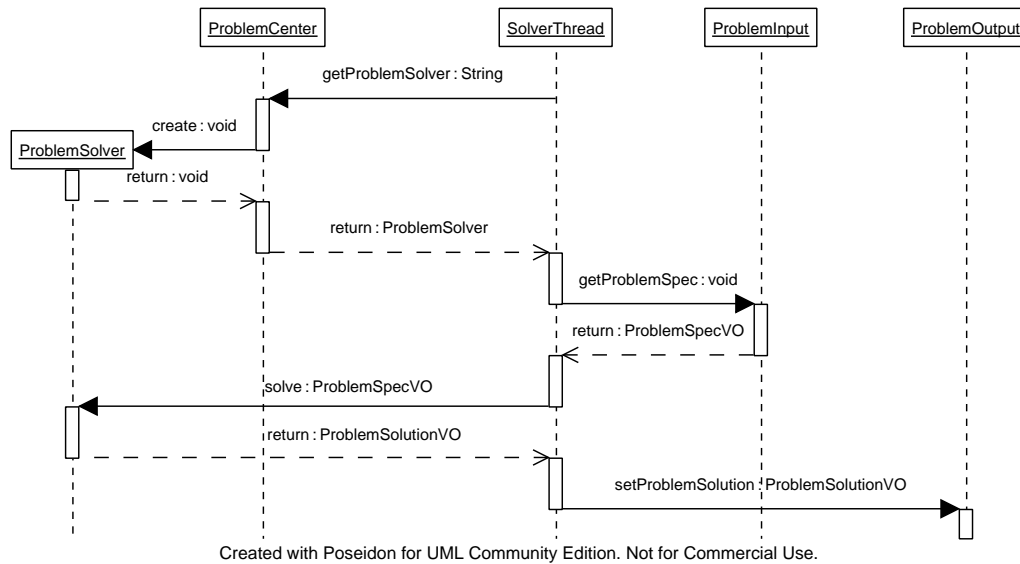


Figura 6.37: La secuencia de resolución de un problema

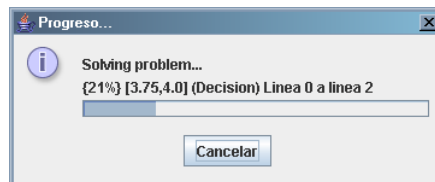


Figura 6.38: El indicador de progreso de las ventanas de problema

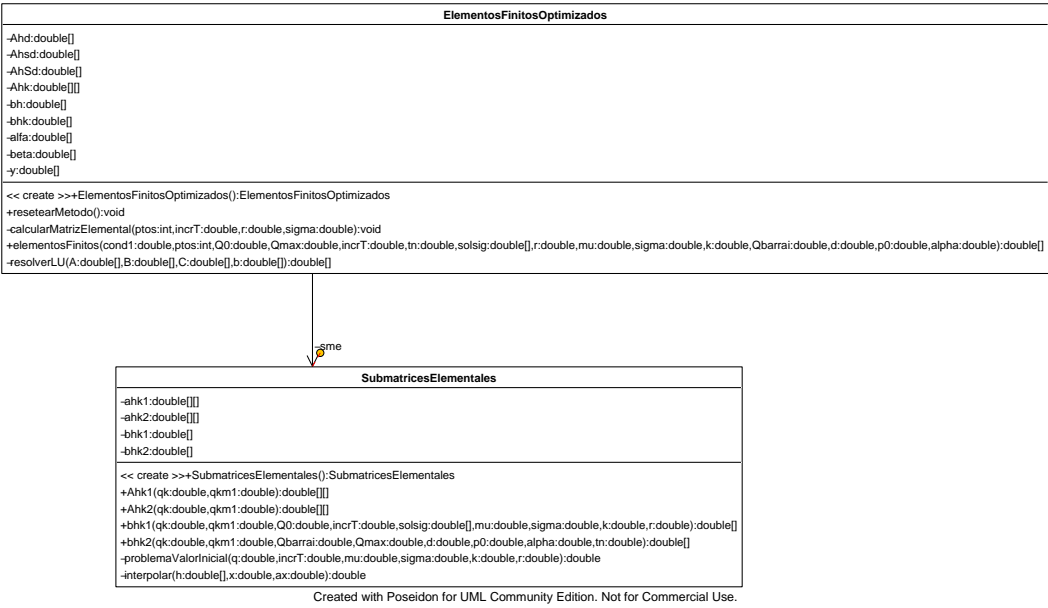


Figura 6.39: Las clases que resuelven la ecuación mediante el método de elementos finitos con características

resolución de los problemas tratados, el de redes de fibra óptica y el de redes de telefonía móvil.

Módulo de toma de decisiones para redes de fibra óptica

Para la construcción del módulo de resolución del problema de redes de fibra óptica se utiliza el conocimiento desarrollado en las secciones 6.1 y 6.2, y se integrarán en un paquete que cumpla los requerimientos especificados por la aplicación general. En concreto, el módulo estará compuesto de dos partes principales, el motor de resolución y la interfaz gráfica.

Motor de resolución : el motor de resolución es el encargado de resolver un problema dado, a partir de su especificación. En él se emplean todas las técnicas anteriormente presentadas, referidas a métodos numéricos y a redes de fibra óptica.

La base del motor de resolución es la resolución de la ecuación en derivadas parciales. Para ello, se definen dos clases, **SubmatricesElementales** y **ElementosFinitosOptimizados**, mostradas en la Figura 6.39.

Para la utilización de estas clases para resolver una EDP se debe crear una instancia de la clase **ElementosFinitosOptimizados** y llamar al método **elementosFinitos**, que recibe todos los parámetros de la ecuación; este método aplica las optimizaciones comentadas en la sección 6.1.2, por lo que aplica el método de elementos finitos completo la primera vez que se llama al método, pero mantiene varias de las estructuras de datos que utiliza para que las llamadas posteriores a ese método se realicen en un tiempo menor. Esto hace que el objeto solamente deba usarse para re-

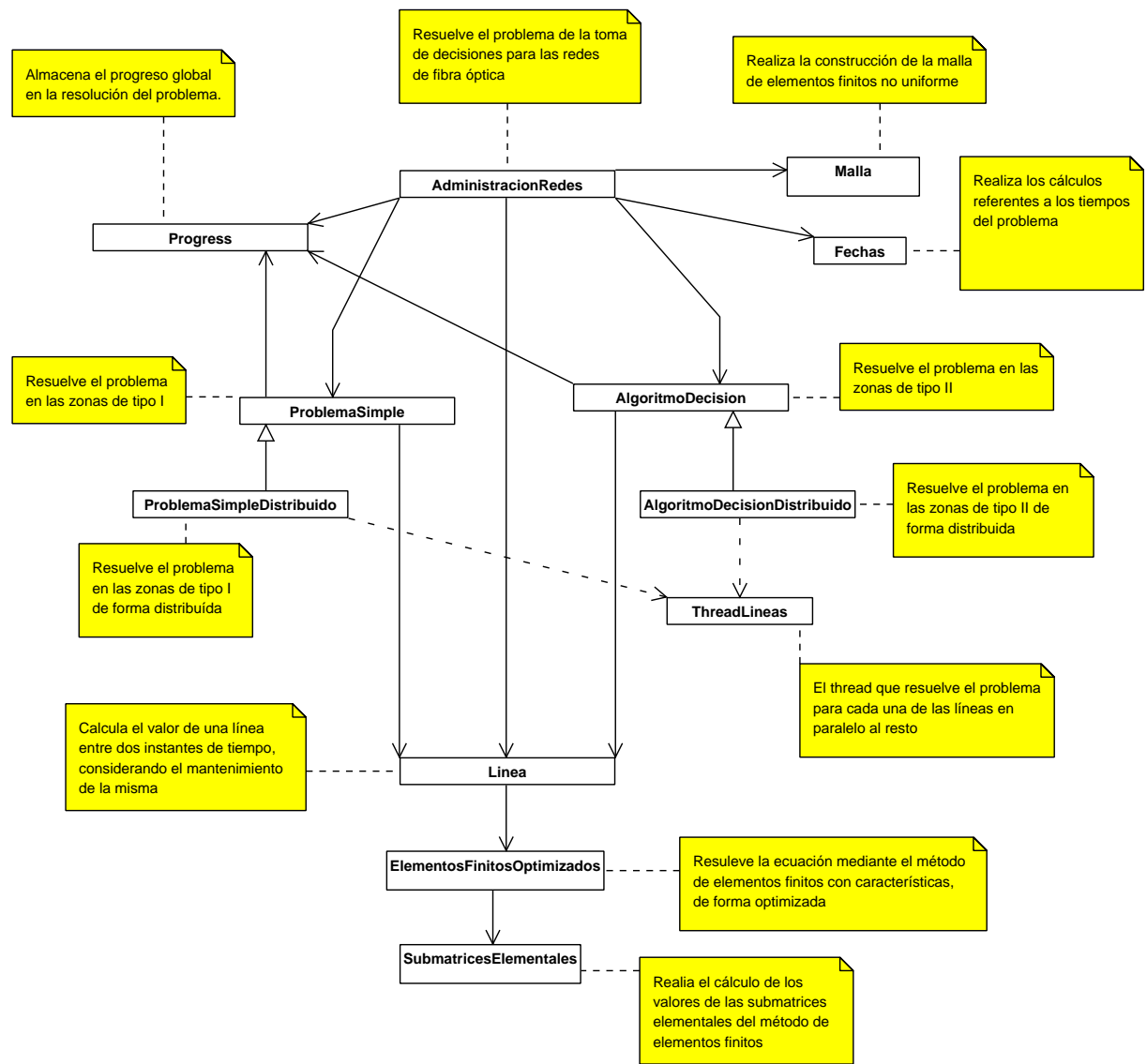
solver una ecuación asociada a un problema, ya que si se resuelve una ecuación con un objeto **ElementosFinitosOptimizados** usado para resolver otra ecuación éste tendrá almacenados datos que habrán sido calculados con parámetros inválidos para la segunda ecuación. Por ello, si se van a resolver ecuaciones con diferentes parámetros se tendrán dos opciones: crear dos instancias diferentes de la clase **ElementosFinitosOptimizados** o bien realizar todas las llamadas de resolución para una de las ecuaciones, posteriormente ejecutar el método **resetearMétodo** del objeto de resolución y por último resolver la segunda ecuación. La operación **resetearMétodo** realiza una limpieza de los datos que almacena el objeto **ElementosFinitosOptimizados** de forma que elimina todo su estado interno, quedando de forma efectiva en la misma situación que la que tenía en el momento inmediatamente posterior a la finalización de su constructor. La clase **SubmatricesElementales** se encarga de la construcción de las submatrices elementales del método de elementos finitos con características, según se ha descrito en los capítulos anteriores; cuando se crea un objeto de la clase **ElementosFinitosOptimizados** automáticamente ese objeto crea una instancia de la clase **SubmatricesElementales** para su uso interno.

Estas clases también serán usadas por el módulo de resolución de problemas de telefonía móvil, ya que tiene que resolver la misma ecuación que el módulo para redes de fibra óptica.

Por otra parte, el resto del módulo de resolución se compone de las clases mostradas en la Figura 6.40. Estas clases se utilizarán de manera similar a la clase **ElementosFinitosOptimizados**; para resolver un problema, se creará una instancia de la clase **AdministraciónRedes**, que internamente creará instancias del resto de clases, bien sea directa o indirectamente. Para realizar la resolución de un problema habrá que llamar al método **resolverProblema** del objeto **AdministraciónRedes** creado. Este método recibe como parámetros todos los datos de entrada del problema, y devuelve una matriz tridimensional que contiene todas las decisiones a tomar, para cada una de las líneas, en cada uno de los tiempos de notificación y cada uno de los posibles valores de Q . Esta matriz será transformada a la hora de presentarla al usuario, para indicar para cada línea y cada instance de notificación el mínimo porcentaje de la capacidad de la línea para el que hay que mejorar dicha línea.

Para calcular esta solución la clase **AdministraciónRedes** crea instancias (directa o indirectamente) de las siguientes clases:

- **Malla**: realiza todos los cálculos referentes a la construcción de las mallas no uniformes que se utilizan en el método de elementos finitos con características. A partir del dominio de Q y del número de puntos que se quiere utilizar para la malla en la variable Q esta clase devuelve un vector que contiene todos los puntos de la malla que el método usará.
- **Fechas**: la clase que realiza las operaciones que involucran cálculos con tiempos. Principal-



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figura 6.40: El motor de resolución de problemas de fibra óptica

mente se encarga de construir los vectores de tiempos relevantes del problema a partir de las frecuencias de mantenimiento, notificación, el parámetro *gamma*, etc.

- **Progress**: al principio de la ejecución, calcula el porcentaje de resolución del problema completo que representa el cálculo de una línea individual en cada una de las zonas; este valor lo utilizan el resto de clases para saber qué porcentaje del problema están resolviendo en cada momento, y el porcentaje global de resolución actual. Este porcentaje también lo almacena la clase **Progress** y se utiliza para informar al usuario del progreso de la resolución, a través de un objeto **ProgressTracker**.
- **ProblemaSimple**: se encarga de resolver el problema de valoración de redes de fibra óptica en las zonas de tipo I (aquellas en las que no hay que tomar decisiones). Para ello, delega el cálculo de las líneas individuales en la clase **Linea**.
- **ProblemaSimpleDistribuido**: como la clase anterior, también resuelve el problema en las zonas de tipo I, pero lo hace de forma distribuida, utilizando el sistema de cálculo distribuido previamente desarrollado. En concreto, cada una de las tareas distribuidas que se ejecutarán resuelve el problema en la zona correspondiente para una línea. Las clases requeridas por el sistema de cálculo distribuido para su utilización se muestran en la figura 6.41, y se discuten más adelante. Estas clases son las que utiliza la clase **ThreadLinea**.
- **AlgoritmoDecision**: esta clase es la que resuelve el problema en las zonas de tipo II, en las cuales hay que invocar el algoritmo de decisión, y que representan las zonas que tienen mayor coste computacional. Como en el caso de la clase **ProblemaSimple**, **AlgoritmoDecision** también delega el cálculo de las líneas individuales en la clase **Linea**.
- **AlgoritmoDecisionDistribuido**: al igual que **AlgoritmoDecision** también se encarga de resolver el problema en las zonas de tipo II, pero lo resuelve de forma distribuida, haciendo uso del sistema de cálculo distribuido desarrollado. En esta resolución también hace uso de las clases de la Figura 6.41, a través de la clase **ThreadLinea**.
- **ThreadLinea**: el *thread* que se encarga de calcular el valor de una línea de forma paralela al resto de líneas. Para ello define el problema del cálculo del valor de una línea en una zona cualquiera (el intervalo entre dos tiempos relevantes) como una subtarea a resolver, creando una clase para la especificación de esa subtarea (**NCMSubSpec**), una clase para la solución de la subtarea (**NCMSubSolution**) y una clase para la resolución de la misma (**NCMSubSolver**). Esta última delega la resolución en la clase **Linea**. Las tres clases se muestran en la Figura 6.41
- **Linea**: calcula el valor de una línea en un intervalo de tiempo. Dispone de dos operaciones, una para calcular el valor teniendo en cuenta el pago de mantenimiento, y otro para resolver la ecuación de la línea sin realizar pagos de mantenimiento (la primera de estas operaciones

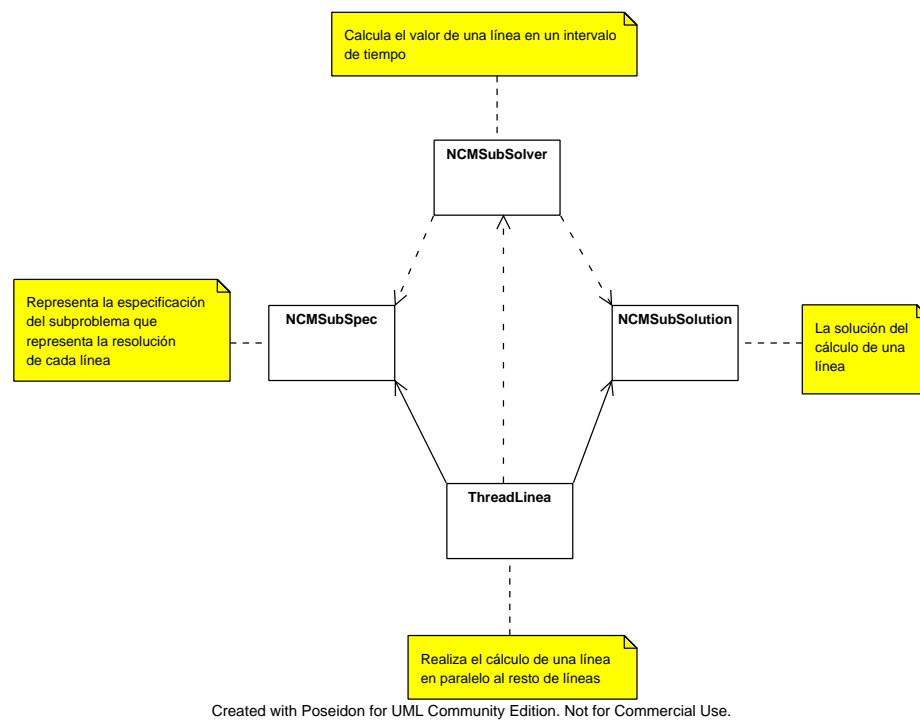


Figura 6.41: Clases requeridas en la resolución distribuida del problema

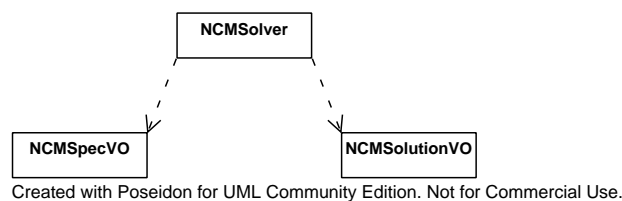


Figura 6.42: Clases para la interfaz del motor de cálculo del módulo de fibra óptica

delega parte del cálculo a la segunda operación).

- **ElementosFinitosOptimizados** y **SubmatricesElementales**: estas clases resuelven la ecuación en derivadas parciales que surge en el problema, como ya se ha comentado anteriormente.

Por último, la clase **NCMSolver** presenta la interfaz de comunicación del módulo con la aplicación, junto con las clases **NCMSpecVO** y **NCMSolutionVO**. La especificación **NCMSpecVO** mantiene todos los datos de entrada del problema; la solución **NCMSolutionVO** contiene las decisiones a tomar para cada una de las líneas y el resolutor **NCMSolver** se encarga de transformar tanto los datos de entrada como los de salida entre los formatos empleados en las clases **NCMSpecVO** y **NCMSolutionVO** y los que utiliza la clase **AdministracionRedes**. Presentando estas tres clases (Figura 6.42) el módulo de resolución define una interfaz que la aplicación sabe cómo invocar cuando necesite ejecutar el caso de uso de resolución del problema.

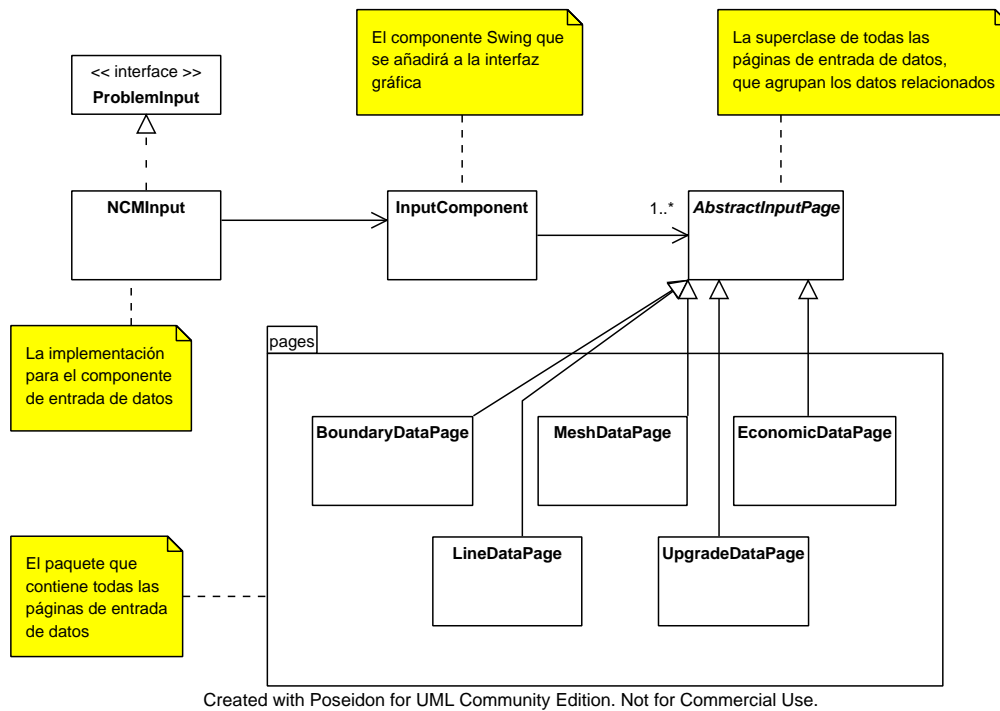


Figura 6.43: Las clases de entrada de datos para la valoración de redes de fibra óptica

Interfaz gráfica : la interfaz gráfica del módulo comprende los componentes que la infraestructura general de la aplicación requiere que se definan, que son una subclase de **ProblemInput**, otra de **ProblemOutput** y una implementación de **ProblemIO**. Esas clases serán las encargadas de realizar la entrada y salida de datos para el problema de las redes de fibra óptica, y componen –junto con las clases **NCMSpecVO**, **NCMSolutionVO** y **NCMSolver**– la implementación de la interfaz requerida por la aplicación.

- **NCMInput:** la clase que se utiliza en la aplicación para introducir los datos del problema de inversiones para redes de fibra óptica (ver Figuras 6.43 y 6.44).

La entrada de datos presenta el problema de tener que manejar una cantidad grande de datos de entrada. En vez de presentárselos todos juntos al usuario, lo que dificultaría su manejo, se dividen los datos en *páginas*. Cada una de estas páginas agrupa los datos que tienen una relación lógica, dividiendo así el conjunto completo de datos de entrada en subconjuntos mucho más manejables por el usuario. Así, se definen las siguientes páginas:

1. Datos de frontera (Figura 6.45): contiene los datos referentes a las fronteras del problema, en concreto, los valores del horizonte de inversión y del valor máximo de Q , además de la opción de resolver el problema de forma distribuída o en la máquina local.
2. Datos de la malla (Figura 6.46): contiene los datos referentes a la malla que se utiliza en el método de elementos finitos (el número de puntos en tiempo y ancho de banda, o el

Figura 6.44: El aspecto del componente de entrada de datos

Figura 6.45: Página de datos de frontera

paso de tiempo y de ancho de banda).

3. Datos financieros (Figura 6.47): contiene los datos referentes al contexto financiero en el que se resuelve el problema, a saber, los parámetros $r, \mu, \sigma, k, P_0, \alpha$ y la longitud de la línea D .
4. Datos de líneas (Figura 6.48): contiene los datos referentes a las diferentes líneas de fibra óptica que se quieren incluir en el problema a resolver. Se indica la frecuencia de mantenimiento y la línea que se quiere anañizar de entre las incluidas en el problema. Además, se dispone para cada una de las líneas de entradas para el ancho de banda y el coste de mantenimiento para esa línea, y un botón para eliminar la línea. Por último, la página incluye un botón para añadir líneas al problema.

Figura 6.46: Página de datos de la malla

Risk free interest rate (r):	<input type="text" value="0,05"/>
Growth rate (mu):	<input type="text" value="0,75"/>
Volatility (sigma):	<input type="text" value="0,95"/>
Market price of risk(k):	<input type="text" value="0,1"/>
Length of the line (D):	<input type="text" value="550"/>
Current spot price (P0):	<input type="text" value="0,9"/>
Decay parameter (alpha):	<input type="text" value="1,4"/>

Figura 6.47: Página de datos financieros del problema

Line to analyze:	<input type="text" value="1"/>
Maintenance frequency:	<input type="text" value="0,083"/>

Line no. 1

Line bandwidth:

Maintenance cost:

Remove line

Line no. 2

Line bandwidth:

Maintenance cost:

Remove line

Line no. 3

Line bandwidth:

Maintenance cost:

Remove line

Line no. 4

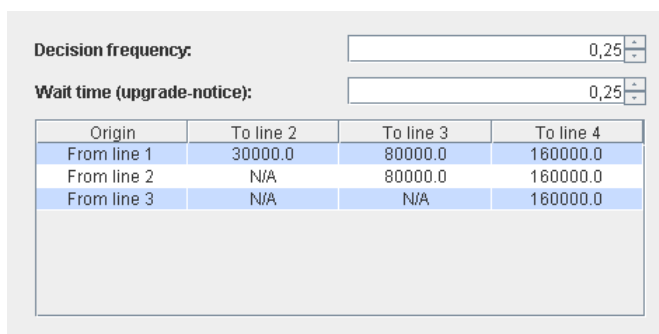
Line bandwidth:

Maintenance cost:

Remove line

New Line

Figura 6.48: Página de datos de las diferentes líneas



Origin	To line 2	To line 3	To line 4
From line 1	30000.0	80000.0	160000.0
From line 2	N/A	80000.0	160000.0
From line 3	N/A	N/A	160000.0

Figura 6.49: Página de datos de mejora de líneas

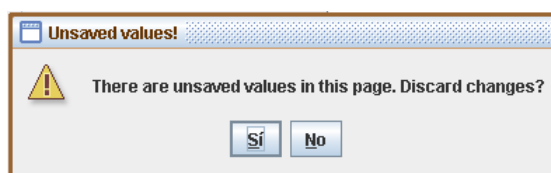


Figura 6.50: Cuadro de diálogo cuando existen datos modificados en un cambio de página

5. Datos de mejora de líneas (Figura 6.49): sirve para introducir los costes de mejora de cada línea a cada una de las líneas de mayor capacidad; estos datos aparecen en forma de tabla. Cada una de las páginas dispone también de un botón para aceptar o cancelar las modificaciones que se han realizado en la misma. Para cambiar entre las diferentes páginas, el usuario deberá seleccionar la página deseada en una lista que aparece en la parte izquierda del componente de entrada y que contiene todas las páginas disponibles. Sin embargo, si el usuario selecciona una página cuando la actual tiene cambios sin confirmar, aparece un cuadro de diálogo que informa de dicha situación y le permite descartar los cambios y pasar a la nueva página seleccionada o volver a la anterior página para guardarlos (Figura 6.50).

- **NCMOutput**: es la implementación de la interfaz **ProblemOutput** para el módulo de toma de decisiones para redes de fibra óptica (figuras 6.51 y 6.52).

La clase **NCMOutput** es responsable de crear y devolver instancias de la clase **SolutionPanel**, que es el componente Swing que muestra la solución al usuario. Este componente incluye una lista desplegable con la que se puede seleccionar la línea para la que se quiere visualizar las decisiones a tomar. Estas decisiones se muestran en una tabla, que contiene una columna para cada una de las líneas a las que se puede mejorar la línea seleccionada y una fila para cada una de las decisiones que hay que tomar.

- **NCMIO**: la clase que realiza el almacenamiento y la carga de los objetos **NCMSpecVO** y **NCMSolutionVO**, mediante los métodos definidos en la interfaz **ProblemIO**, e implementa la funcionalidad de los casos de uso de cargar y almacenar datos de disco, e imprimirlos.

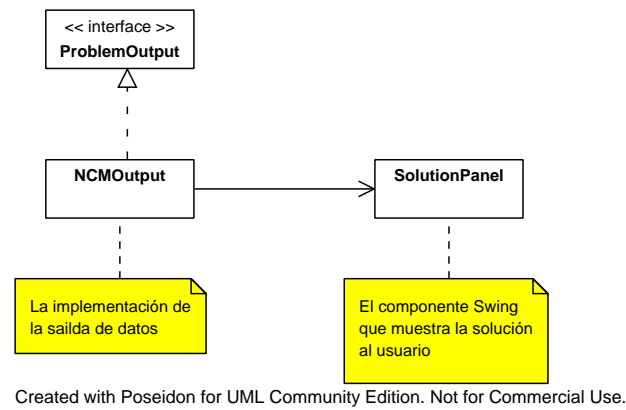


Figura 6.51: Las clases de salida de datos para la valoración de redes de fibra óptica

Decision for line: 1				
Time	To line 2	To line 3	To line 4	
0.0	128,62 %	273,31 %	787,78 %	
0.25	128,62 %	289,39 %	836,01 %	
0.5	144,69 %	321,54 %	900,32 %	
0.75	160,77 %	353,7 %	980,71 %	
1.0	192,93 %	401,93 %	1077,17 %	
1.25	225,08 %	450,16 %	1189,71 %	
1.5	289,39 %	530,55 %	1318,33 %	
1.75	385,85 %	627,01 %	1495,18 %	
2.0	691,32 %	755,63 %	1688,1 %	
2.25	-	1077,17 %	1945,34 %	
2.5	-	1816,72 %	2282,96 %	
2.75	-	-	2958,2 %	
3.0	-	-	4437,3 %	
3.25	-	-	9839,23 %	
3.5	-	-	-	
3.75	-	-	-	
4.0	-	-	-	
4.25	-	-	-	
4.5	-	-	-	

Figura 6.52: El aspecto del componente de la solución del problema

Módulo de toma de decisiones para redes de telefonía móvil

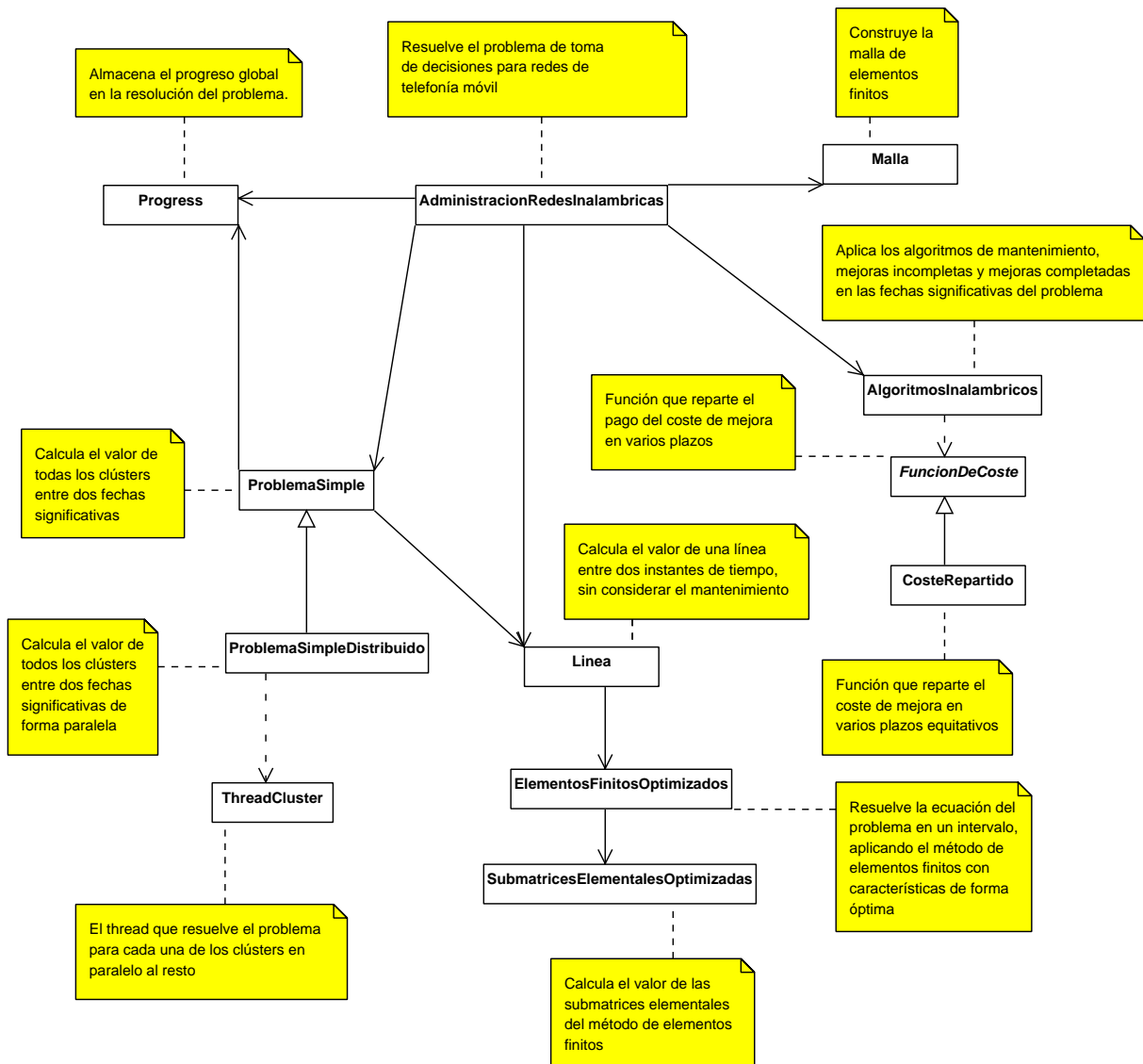
El módulo para la resolución de problemas de toma de decisiones en redes de telefonía móvil es muy similar al módulo de toma de decisiones en redes de fibra óptica. Al igual que éste, está formado por dos partes, el motor de resolución y la interfaz gráfica.

Motor de resolución : el motor de resolución está organizado de forma similar al del módulo de resolución de problemas de fibra óptica. La base de la resolución son las clases utilizadas en ese módulo para la resolución de la ecuación en derivadas parciales mediante el método de elementos finitos con características, `ElementosFinitosOptimizados` y `SubmatricesElementales`. Estas clases ya han sido descritas anteriormente, y no se detallarán más en esta sección.

Además de esas dos clases comunes, también se reutilizan las clases `Malla` y `Linea`. La clase `Malla` se reutiliza porque la malla de elementos finitos usada en la valoración de redes de telefonía móvil es la misma que la de la valoración de redes de fibra óptica. La clase `Linea` se utiliza parcialmente: solamente es útil para este problema la operación para resolver la ecuación en derivadas parciales sin tener en cuenta los pagos de mantenimiento, ya que las diferencias en el algoritmo de valoración de ambos tipos de redes hace que el pago de los plazos de mantenimiento de las redes de telefonía se realice en otra clase.

El conjunto de las clases que se utilizan en el motor de resolución se muestra en la Figura 6.53.

- **Malla**: sirve para construir la malla de elementos finitos no uniforme, de la misma forma que se hacía para las redes de fibra óptica.
- **ElementosFinitosOptimizados** y **SubmatricesElementales**: realizan el cálculo de la ecuación de la forma ya comentada.
- **Linea**: calcula el valor de un clúster individual en un período de tiempo sin tener en cuenta los pagos de mantenimiento. Para ello se basa en la clase `ElementosFinitosOptimizados`.
- **ProblemaSimple**: resuelve el conjunto de ecuaciones de todos los clústers para calcular su valor entre dos fechas significativas, sin tener en cuenta el mantenimiento, mediante la clase `Linea`. Mediante esta clase se calcula el valor de todos los clústers en todos los estados de terminación de sus respectivas mejoras, si hay mejoras pendientes.
- **ProblemaSimpleDistribuido**: al igual que `ProblemaSimple`, calcula el valor de los clústers entre dos fechas significativas, pero realiza el cálculo de forma distribuida: el valor de cada clúster se calcula en un nodo diferente, y los resultados se integran de forma local. Para el cálculo del valor de cada uno de los clústers esta clase hace uso de la clase `ThreadLinea`.
- **ThreadCluster**: un *thread* que calcula el valor de un clúster entre dos fechas significativas. Para ello, crea un objeto de la clase `WNCMSubSpecVO`, que define el problema de calcular el valor de un



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figura 6.53: Clases del motor de resolución para redes de telefonía móvil

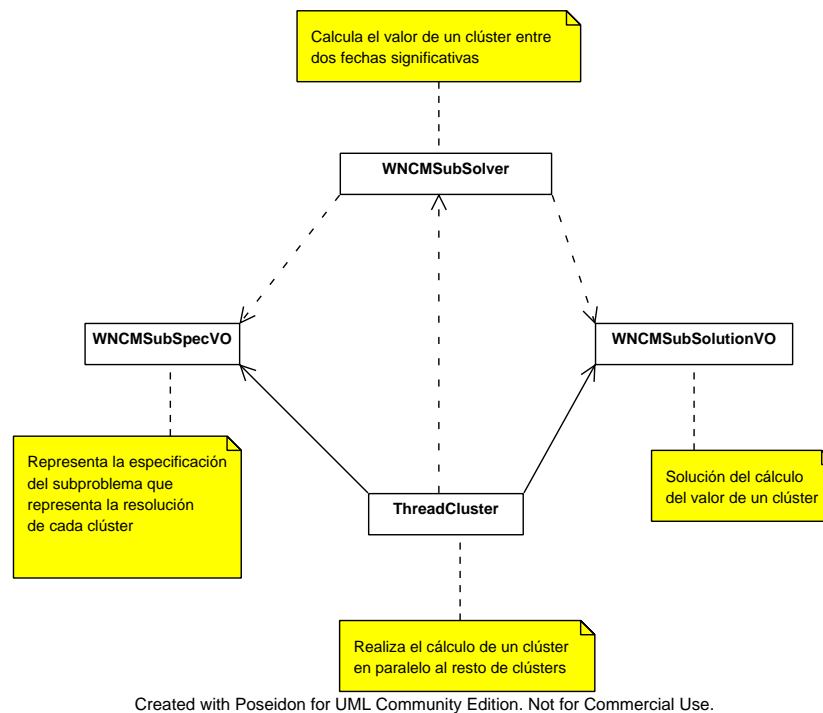


Figura 6.54: Clases para la resolución dsitribuída del problema

único clúster entre dos fechas significativas, y a través del sistema de cálculo distribuído resuelve ese problema en otro ordenador, obteniendo finalmente un objeto `WNCMSubSolutionVO`, que contiene el valor inicial del clúster. Estas clases se muestran en la Figura 6.54.

- **AlgoritmosInalambricos:** esta clase aplica todos los algoritmos de las redes de telefonía móvil en las fechas significativas, definidos en la sección 3.2.3. Mediante estos algoritmos se actualizan los pagos de mantenimiento, se actualizan los valores de los clústers con mejoras sin completar y se toman las decisiones de los clústers que han completado los cálculos de mejoras.
- **FuncionDeCoste:** esta interfaz define el contrato que deben cumplir las clases que se quieran utilizar para calcular el valor de los diferentes plazos que se han de pagar cuando se fragmenta el pago de la cuota de mejora de un clúster a lo largo del tiempo. El tener una interfaz común para todas las funciones hace que el añadir nuevas formas de pago no requiera más esfuerzo que el necesario para codificar dicha función, tratándose todas ellas de un modo uniforme.
- **CosteRepartido:** una implementación de `FuncionDeCoste` que divide el coste total de la mejora en varios plazos iguales, ajustados para tener en cuenta la tasa de interés libre de riesgo.
- **AdministracionRedesInalambricas:** resuelve el problema de la toma de decisiones para inversiones en redes de telefonía móvil. Para ello recibe todos los parámetros que definen el problema y devuelve una matriz tridimensional con las decisiones a tomar para cada uno de

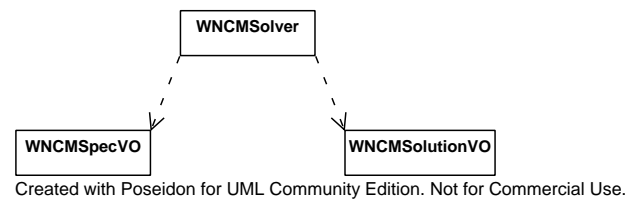


Figura 6.55: Las clases de la interfaz del módulo para redes de telefonía móvil

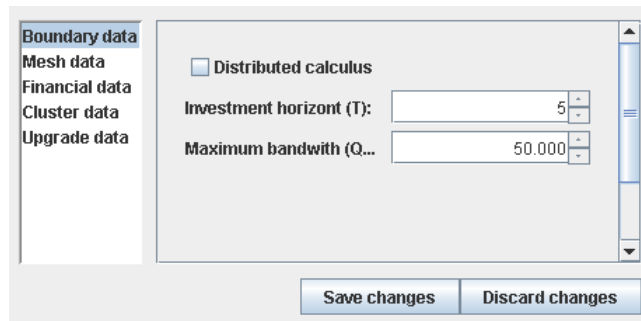


Figura 6.56: El componente de entrada de datos para redes de telefonía móvil

los clústers en cada una de las fechas de notificación para cada uno de los posibles valores de Q . Esta matriz deberá ser transformada para que el usuario pueda visualizar los resultados de forma cómoda, de forma similar a la empleada en el módulo de redes de fibra óptica.

Además de las clases que forman el motor de cálculo para el problema en cuestión, la aplicación de resolución de problemas requiere que el módulo implemente las interfaces con las que ambos se comunican, (`ProblemSpecVO`, `ProblemSolutionVO` y `ProblemSolver`). Las implementaciones en este módulo son, respectivamente, `WNCMSpecVO`, `WNCMSolutionVO` y `WNCMSolver`, que se muestran en la figura 6.55.

Interfaz gráfica : la interfaz gráfica del módulo se toma de decisiones para redes de telefonía móvil es muy similar a la interfaz para redes de acceso a Internet. Como ella, está formada por tres componentes principales, impuestos por la infraestructura genérica de la aplicación: un `ProblemInput`, un `ProblemOutput` y un `ProblemIO`.

- **WNCMInput:** la subclase de `ProblemInput` para este problema. A partir de ella se obtiene el componente Swing que se incluye en la interfaz gráfica para realizar la entrada de datos (este componente se muestra en la Figura 6.56). La estructura de clases del componente de entrada se puede observar en la Figura 6.57; se aprecia que es casi exactamente igual que la estructura del componente de entrada de datos del problema de toma de decisiones en redes de acceso a Internet. Como en el caso de las redes de fibra óptica, el conjunto de datos que define un

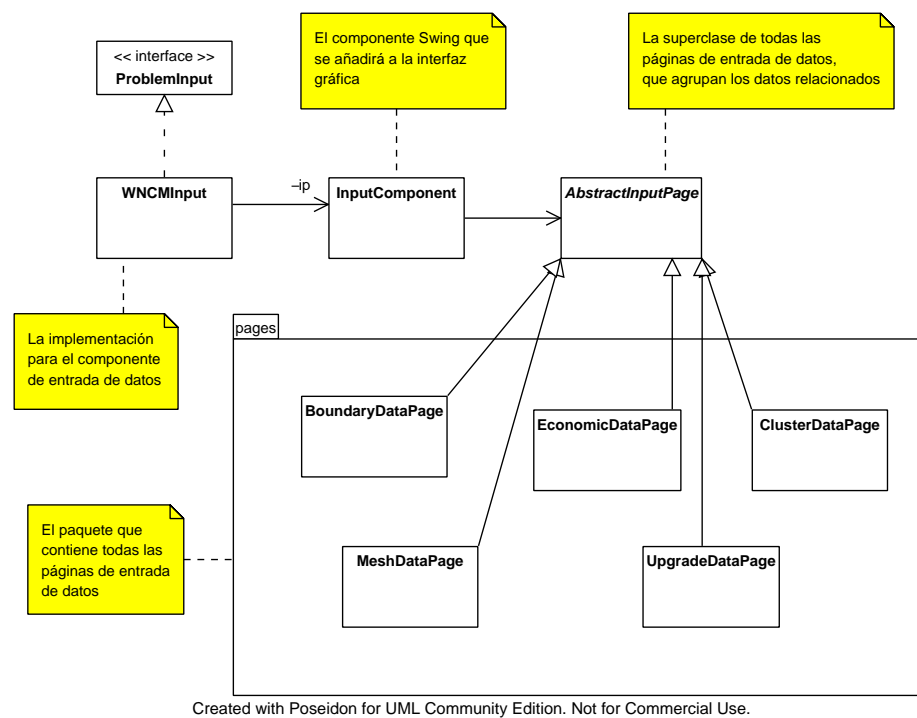


Figura 6.57: La estructura de clases del componente de entrada de datos

☐ Distributed calculus

Investment horizon (T):

Maximum bandwidth (Q...):

Figura 6.58: Página para los datos de frontera

problema es demasiado extenso para manejarlo como una sola unidad, de forma que se divide en subconjuntos formados por datos relacionados entre sí. Cada uno de los conjuntos se edita en una *página*. Las páginas disponibles son las mismas que en el problema de redes de fibra óptica:

1. Datos de frontera (Figura 6.58): para la introducción de los datos relacionados con las fronteras de resolución de la EDP, junto con la opción de resolver el problema de forma distribuída.
2. Datos de la malla (Figura 6.59): donde se introducen los datos referentes a la malla de elementos finitos utilizada en el problema.
3. Datos financieros (ver Figura 6.60): contiene los datos del mercado en el momento actual $(r, \mu, k, \sigma, P_0, \alpha)$.
4. Datos de los clústers (Figura 6.61): en la que se introducen los datos relacionados con los

Figura 6.59: Página para los datos de la malla

Figura 6.60: Página para los datos financieros

diferentes tipos de clústers que se consideran, con su capacidad de transmisión y su coste de mantenimiento.

5. Datos de mejora (ver Figura 6.62): donde se introduce el número de portadores por estación base, el coste de cada portador, la frecuencia de las decisiones de mejora y el tiempo de espera entre los tiempos de notificación y los de mejora (γ).

Además de las páginas, en el componente de entrada hay botones para confirmar o descartar los cambios realizados en la página actual; como en el componente de entrada para el problema de las redes de acceso a Internet, cuando se pasa a una página desde otra que tiene cambios sin confirmar, se pregunta al usuario si desea descartarlos o guardarlos (ver Figura 6.50).

- **WNCMOutput**: la implementación para este módulo de la interfaz **ProblemOutput**. Esta clase provee un método para obtener el componente Swing que muestra el resultado de los cálculos en la interfaz gráfica. El aspecto de dicho componente se muestra en la Figura 6.63, y su diagrama de clases en la figura 6.64. Dichos resultados se muestran en forma de tabla, como en el caso de las redes de acceso a Internet. Existe una tabla para cada uno de los clústers del problema; cada una de estas tablas tiene una columna para cada clúster de mayor capacidad, y una fila para cada fecha de notificación. En cada una de las celdas se muestra el mínimo porcentaje de capacidad del clúster utilizado para el que se debe mejorar al clúster correspondiente a la columna de la celda, en la fecha de decisión correspondiente a la fila de la celda.
- **WNCMIO**: la clase que realiza la entrada/salida de datos en disco: provee de métodos para

Cluster no. 1

Cluster bandwidth: 2.376

Maintenance cost: 738.000

Remove cluster

Cluster no. 2

Cluster bandwidth: 5.580

Maintenance cost: 858.000

Remove cluster

Cluster no. 3

Cluster bandwidth: 8.928

Maintenance cost: 978.000

Remove cluster

New Cluster

Figura 6.61: Página para los datos de los diferentes clústers

Decision frequency: 0,083

Wait time (gamma): 0,333

Carriers per cell site: 20

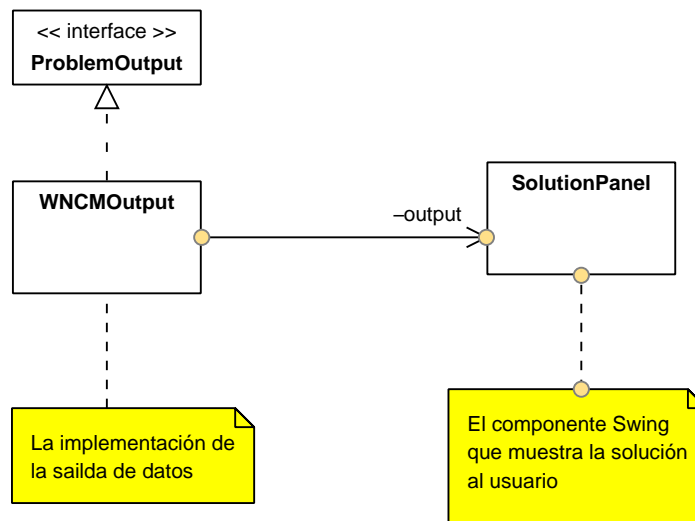
Carrier cost: 150.000

Figura 6.62: Página para los datos de mejora de los clústers

Decision for cluster: 1

Time	Add 1 carrier(s)	Add 2 carrier(s)
3.7499999999999996	18,31 %	35,92 %
3.8333333333333333	21,36 %	35,92 %
3.9166666666666665	21,36 %	40,14 %
3.9999999999999996	21,36 %	40,14 %
4.0833333333333333	21,36 %	40,14 %
4.1666666666666667	24,65 %	44,6 %
4.25	24,65 %	44,6 %
4.3333333333333333	28,17 %	49,3 %
4.4166666666666667	31,92 %	54,23 %
4.5	35,92 %	64,79 %
4.5833333333333333	44,6 %	82,39 %

Figura 6.63: Componente Swing que muestra los resultados del problema



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figura 6.64: Clases del componente de salida de datos

guardar, cargar e imprimir datos de entrada y resultados, de la misma forma que lo hace la clase **NCMIO** para el problema de las redes de fibra óptica.

Capítulo 7

Resultados obtenidos

En este capítulo se presentan los resultados obtenidos acerca de las secciones teóricas presentadas anteriormente. Cada una de las secciones está dedicada a un área del problema, comenzando por los resultados del método numérico.

7.1. Resultados numéricos

En esta sección se presentan los resultados obtenidos con respecto al método numérico empleado y la aproximación de la solución de las ecuaciones en derivadas parciales que aparecen en el problema.

En primer lugar, se muestra la comparación de las soluciones¹ obtenidas cuando se utilizan mallas uniformes y las obtenidas al usar mallas no uniformes. Como se comentó en su momento, las mallas no uniformes presentan la ventaja de necesitar menos puntos que las uniformes para obtener la misma precisión, al aprovechar el conocimiento que se tiene acerca de la solución. Para realizar esta comparación, se muestran las figuras 7.1 y 7.2. En ambas figuras se comparan las soluciones obtenidas utilizando dos mallas no uniformes, de 10000 y 16000 puntos en la variable Q , respectivamente, en colores azul y rojo, con las calculadas usando mallas uniformes de 10000 y 16000 puntos, en colores magenta y negro. La Figura 7.1 muestra la solución en el dominio completo de Q , donde no se aprecian diferencias entre las cuatro soluciones. En cambio, al observar en la Figura 7.2 en detalle la zona donde la derivada cambia su valor, las diferencias se vuelven apreciables. En efecto, la peor aproximación es la obtenida con la malla uniforme de 10000 puntos (color negro). Esta aproximación es muy pobre, y oscila a ambos lados de la solución real de forma incontrolada, resultando inservible de cara a la resolución del problema. La aproximación mejora sustancialmente si se aumenta el número de puntos a 16000 (color magenta). Sin embargo, la aproximación obtenida sigue siendo muy deficiente, y todavía inservible, debido a las

¹las referencias a las soluciones deberán ser tomadas como referencias a las aproximaciones de las soluciones, ya que se está empleando un método numérico que no proporciona la solución exacta. Por motivos de brevedad, en la mayoría de los casos se aludirá a dichas aproximaciones como soluciones

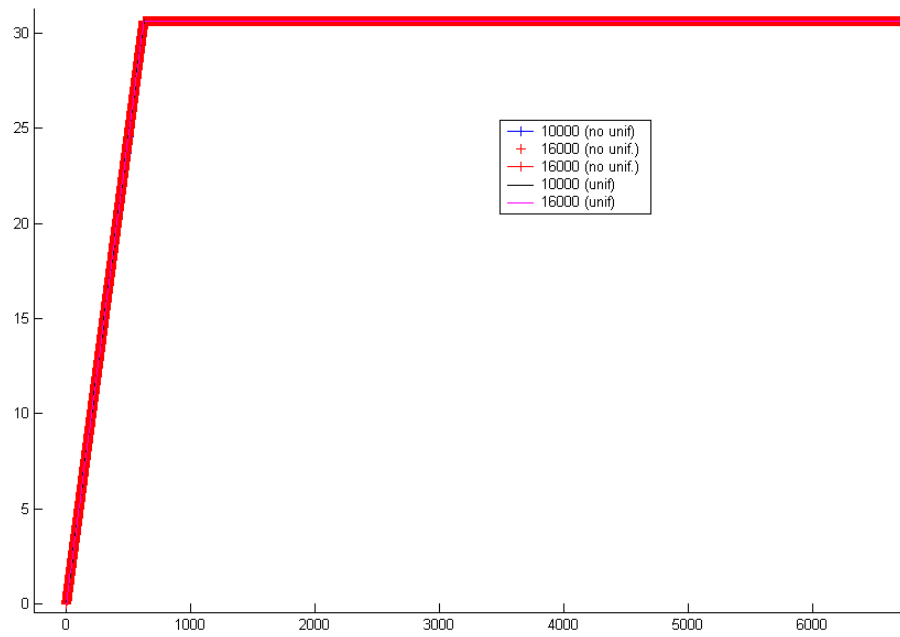


Figura 7.1: Soluciones obtenidas con diferentes mallas

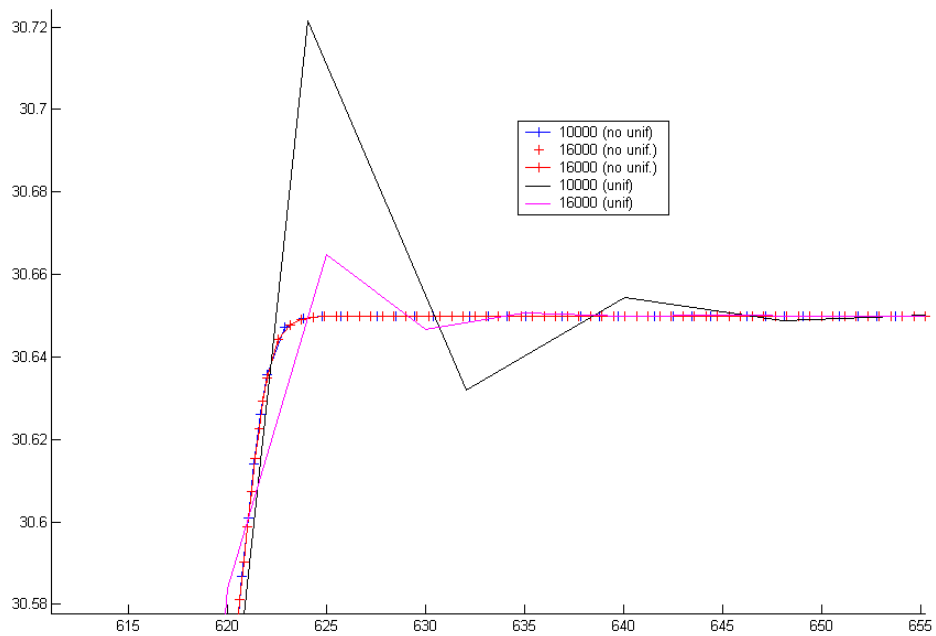


Figura 7.2: Soluciones obtenidas con diferentes mallas (detalle)

oscilaciones ya comentadas (que en este caso son menores que en el anterior). De esta forma, se deberían utilizar un número de puntos del orden de 100000 para comenzar a obtener resultados aprovechables. Si se utiliza una malla no uniforme, las soluciones obtenidas mejoran sustancialmente. En color azul se muestra la aproximación conseguida mediante la utilización de una malla no uniforme de 10000 puntos: dicha solución describe una curva suave, en la que no se aprecia ninguna clase de oscilaciones que invaliden la aproximación. Si se compara el valor obtenido de esta forma con el valor obtenido con la malla uniforme del mismo número de puntos, se observa la ganancia de calidad que proporciona la distribución no homogénea de los puntos. En color rojo se observa el efecto que tiene el aumentar el número de puntos de dicha malla de 10000 a 16000: se puede comprobar que ambas soluciones están muy próximas entre sí; esto indica que con ese número de puntos la aproximación está cerca del valor real de la solución.

Así, mediante ambas figuras se demuestra la corrección de la elección en la clase de malla utilizada.

En segundo lugar, se muestra la convergencia de la solución según se aumenta el número de puntos utilizados. En las ecuaciones en derivadas parciales presentes en los problemas existen dos variables, tiempo y ancho de banda demandado. Así, las soluciones de dichas ecuaciones tienen dos dimensiones (pese a que en la mayoría de situaciones solamente se muestran los resultados en la variable Q , debido a que no interesa la evolución en la variable t a la hora de la toma de decisiones). Así, la densidad de la malla puede modificarse en cada una de las variables del problema de forma independiente (a priori). Sin embargo, ambas densidades no son completamente independientes si se quieren conseguir buenas aproximaciones: el método de elementos finitos con características empleado tiene una convergencia con orden

$$O\left(\frac{\Delta Q^2}{\Delta t} + \Delta Q + \Delta t\right)$$

siendo ΔQ la separación entre puntos en la variable Q y Δt la separación entre puntos en la variable t . Aunque usando mallas no uniformes no se puede hablar de una separación fija entre puntos, sí se pueden aplicar las conclusiones al número de puntos utilizado. Así, el orden de convergencia depende tanto del número de puntos en Q como en t : si ΔQ se hace mucho más pequeño que Δt , los beneficios no serán apreciables debido al término Δt que se suma; si Δt se hace mucho más pequeño que ΔQ , no sólo los efectos no serán apreciables, sino que la convergencia será peor al aumentar el término $\frac{\Delta Q^2}{\Delta t}$. Por esta razón, si se quieren obtener resultados correctos se ha de aumentar siempre el número de puntos en ambas dimensiones simultáneamente. En consecuencia, siempre que se hace referencia al aumento de puntos en la variable Q , hay un aumento implícito del número de puntos en t (esto hace que el tiempo de cálculo aumente de una forma notable cuando se aplican aumentos no demasiado grandes en el número de puntos de la malla). Teniendo esto en cuenta, se está en condiciones de interpretar correctamente las figuras 7.3 y 7.4.

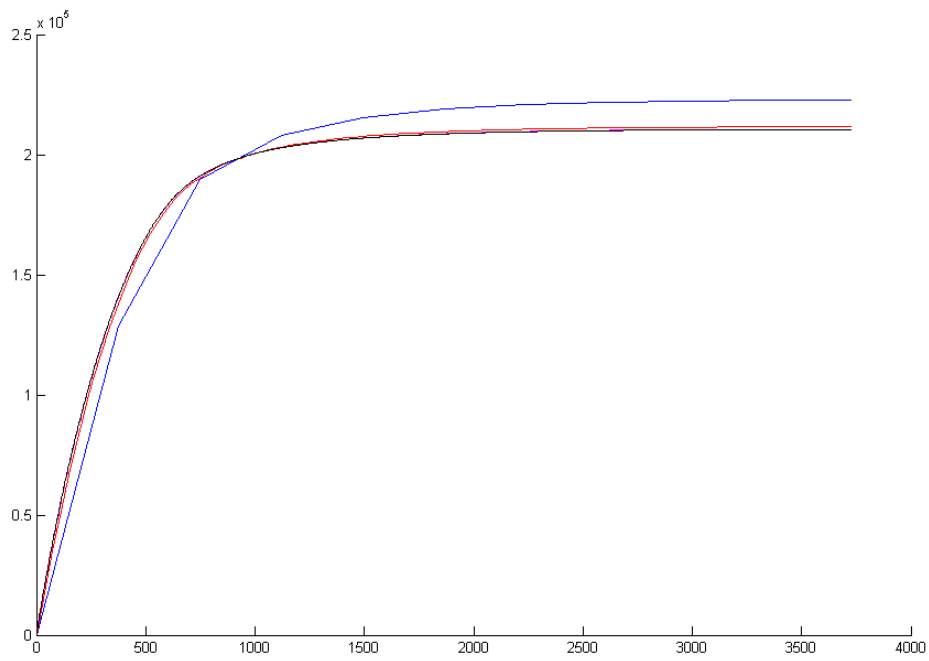


Figura 7.3: Convergencia de la solución al aumentar la densidad de la malla

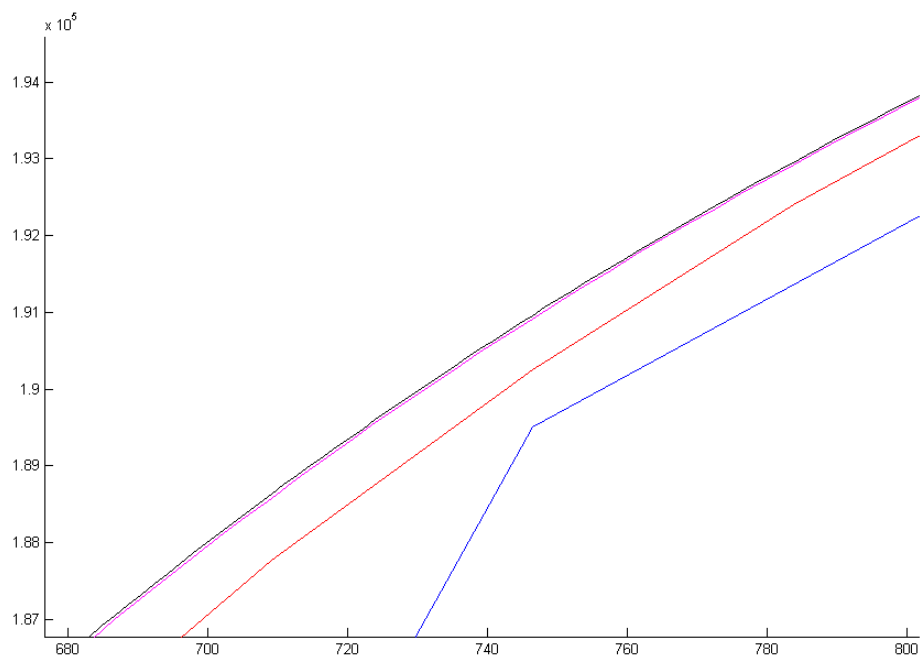


Figura 7.4: Convergencia de la solución al aumentar la densidad de la malla (detalle)

Color	Número de puntos
Azul	11
Rojo	101
Magenta	1001
Negro	2001

Cuadro 7.1: Número de puntos de cada malla

En ambas figuras se observan las aproximaciones obtenidas mediante cuatro mallas diferentes; en la Tabla 7.1 se muestra el número de puntos utilizado para cada uno de los colores.

En la Figura 7.3 se aprecia claramente la diferencia entre la peor solución (azul) y el resto de soluciones; además, se adivina la diferencia entre la siguiente peor solución (rojo) y las dos mejores. Esta diferencia se ve claramente en la Figura 7.4. También es posible ver que las soluciones magenta y negra son muy próximas entre si: aumentando el número de puntos en ancho de banda (y en tiempo) las soluciones obtenidas convergen a la solución de la ecuación.

Por último, es relevante comentar que al introducir los pagos de mantenimiento se introducen discontinuidades en la solución de la ecuación. Estas discontinuidades siempre son malas de cara a la resolución numérica; en el presente caso, las discontinuidades (mostradas en la Figura 6.1 de la sección 6.1.1) no invalidan las aproximaciones numéricas, solamente hacen que la convergencia sea más lenta.

7.2. Resultados para decisiones sobre redes de fibra óptica

7.2.1. Parámetros empleados

En las tablas 7.2, 7.3 y 7.4 se muestran los valores de los parámetros que se han usado para obtener los resultados para redes de fibra óptica. Estos son los parámetros por defecto, usados en todas las pruebas, a no ser que se indique lo contrario. Adicionalmente, son los mismos datos de entrada que utilizan los autores de [4].

7.2.2. Resultados obtenidos

En primer lugar, se muestran los resultados del caso base, en la tabla 7.5. En dicha tabla se muestra el mínimo porcentaje de ocupación de la línea para el que se debe mejorar. En cada fila se observa la fecha de notificación, comenzando en $t = 0$ (instante actual) y terminando en $t = 2$ (no se muestran los resultados hasta el horizonte de inversión $t = 5$). En cada columna se muestran los resultados para cada una de las líneas; no se muestran todas las posibles mejoras, sino solamente las mejoras a la siguiente línea. El primer efecto que se debe observar es el crecimiento de los porcentajes a lo largo del tiempo. Esto es debido a

Horizonte de inversión (T)	5 años
Tasa de interés (r)	0.05
Tasa de crecimiento (μ)	0.75
Volatilidad (σ)	0.95
Precio del mercado de riesgo (k)	0.10
Precio de venta del minuto (P_0)	0.90
Factor de decaimiento (α)	1.40
Longitud de la línea (d)	550

Cuadro 7.2: Datos de entrada para el problema de las redes de acceso a Internet

Línea	Capacidad	Coste de mantenimiento anual por milla
1 (OC-12)	622	2.4
2 (OC-48)	2488	18
3 (OC-192)	9952	48
4 (OC-768)	39808	96

Cuadro 7.3: Datos de las líneas usadas

De línea	a OC-48	a OC-192	a OC-768
OC-12	30000	80000	160000
OC-48	N/A	80000	160000
OC-192	N/A	N/A	160000

Cuadro 7.4: Costes de mejora de las líneas

Tiempo	OC-12 a OC-48	OC-48 a OC-192	OC-192 a OC-768
0.00	104.5 %	84.41 %	64.11 %
0.25	112.54 %	88.42 %	66.22 %
0.50	123.79 %	93.65 %	68.93 %
0.75	139.87 %	100.88 %	72.55 %
1.00	160.77 %	110.13 %	77.17 %
1.25	191.32 %	122.59 %	83.10 %
1.50	241.16 %	139.47 %	90.64 %
1.75	334.41 %	162.78 %	100.18 %
2.00	607.72 %	196.95 %	112.34 %

Cuadro 7.5: Toma de decisiones para los dos primeros años del caso base

que a medida que pasa el tiempo, hay menos oportunidades de amortizar el hardware (ya que el beneficio percibido es menor si el intervalo de tiempo es menor); así, cuanto más tarde se tome la decisión más alta tiene que ser la demanda, para poder obtener los beneficios suficientes para que resulte rentable el haber mejorado la línea. Este razonamiento tiene factores adicionales, como son los costes decrecientes de mejora, los costes fijos de mantenimiento, la función R decreciente. . . Sin embargo, la situación existente es la ya expuesta: con decisiones más tardías, hace falta una mayor demanda para amortizar las mejoras. En cuanto a los valores obtenidos, claramente son superiores a los que parecen guiar las inversiones en las empresas (mejorar cuando el porcentaje de utilización llegue al 50 %); en efecto, indican que es mejor esperar a que la línea sobrepase su capacidad máxima de transmisión (en el caso de las líneas OC-12) antes de mejorarla; esto no es completamente realista, debido al efecto que la congestión tiene sobre el beneficio (los efectos de la congestión se comentan más adelante). Así, la práctica común de mejorar la infraestructura cuando la demanda llega a la mitad de la capacidad máxima de la red conduce claramente a una situación de sobredimensionamiento de la red, en la que se está adquiriendo un hardware que no va a producir los beneficios que lo amorticen (como ya se explicó en la Introducción del presente Proyecto Fin de Carrera).

Los resultados de la tabla 7.5 no tienen en cuenta los efectos de la congestión en la línea de fibra óptica cuando la demanda sobrepasa la capacidad máxima de la línea. Esto hace que los valores que se han mostrado en dicha tabla no sean completamente realistas, ya que en muchos casos recomiendan no mejorar la línea hasta que la demanda sobrepase la capacidad máxima de la línea en un gran porcentaje, pero no consideran el efecto de penalización que aparece debido a la congestión de las líneas. Por ello, se incluye en la aplicación la posibilidad de tener en cuenta dicha congestión; la forma de hacerlo se estableció en el capítulo 2. Los resultados de la inclusión de este efecto en el modelo se muestran en la tabla 7.6. Estos resultados muestran una reacción a la inclusión de la congestión en el motor de resolución; como era de esperar, los porcentajes obtenidos como demanda mínima para mejorar la línea son más bajos que los obtenidos sin tener en cuenta la penalización por congestión (entre un 15 y un 30 por ciento). Esto es algo esperado, ya que al eliminarse los beneficios cuando la demanda supera el 120 % de la capacidad máxima de la línea, los porcentajes de mejora decrecen para evitar dicha pérdida de beneficios. Así, el motor de cálculo continúa maximizando el beneficio en la nueva situación, intentando mantener la demanda dentro de los límites sin congestión. Esto proporciona unos resultados mucho más realistas que los calculados sin la penalización por sobrecarga de la línea; por ello, siempre debería incluirse este factor en los cálculos.

Tiempo	OC-12 a OC-48	OC-48 a OC-192	OC-192 a OC-768
0.00	75.56 %	62.29 %	48.83 %
0.25	80.38 %	65.11 %	50.24 %
0.50	90.03 %	68.72 %	52.04 %
0.75	101.28 %	73.55 %	54.56 %
1.00	118.97 %	79.98 %	57.67 %
1.25	151.12 %	88.826 %	61.69 %
1.50	273.31 %	100.88 %	66.82 %
1.75	—	118.97 %	73.25 %
2.00	—	150.72 %	81.39 %

Cuadro 7.6: Efecto de la congestión de las líneas en las decisiones

Horizonte de inversión (T)	5 años
Δt_{obs}	1 mes
Número de estaciones en el clúster	20
Tiempo de espera (γ)	4 meses
Tasa de interés (r)	0.04
Tasa de crecimiento (μ)	0.3
Volatilidad (σ)	0.65
Precio del mercado de riesgo (k)	0.03
Precio de venta del minuto (P_0)	8015
Factor de decaimiento (α)	0.05

Cuadro 7.7: Datos de entrada para el problema de las redes de telefonía móvil

7.3. Resultados para decisiones sobre redes de telefonía móvil

7.3.1. Parámetros empleados

Para calcular los resultados que se incluyen en esta sección se han utilizado los datos mostrados en las tablas 7.7, 7.8, 7.9 y 7.10, excepto en los casos en los que se indique lo contrario; estos son los datos que se utilizan en [5].

7.3.2. Resultados obtenidos

La primera tabla de resultados (7.11) muestra los resultados obtenidos utilizando mallas de elementos finitos cada vez más finas, para diferentes intervalos entre decisiones de mejora. Cada resultado se muestra

Número de portadores	Capacidad del clúster	Coste de mantenimiento anual
1	2376	738000
2	5580	858000
3	8928	978000

Cuadro 7.8: Datos de los clústers usados

De x portadores	a 2 portadores	a 3 portadores
1	3000000	6000000
2	N/A	3000000

Cuadro 7.9: Costes de mejora de los clústers

como el mínimo porcentaje de capacidad utilizado para el que se debe realizar la mejora; este porcentaje se muestra para decisiones mensuales, trimestrales, semestrales y anuales, y para diferentes niveles de refinamiento de la malla. El nivel de refinamiento inicial toma 250 puntos para la variable tiempo y 70 para la demanda de capacidad, y cada nivel posterior duplica el número de puntos tanto en tiempo como en ancho de banda. Se observa que al aumentar el intervalo entre tiempos de notificación la decisión de mejorar el clúster se ha de tomar cada vez más temprano. Intuitivamente, este efecto aparece debido a que debido a que van a existir menos decisiones, es mejor aumentar la capacidad antes, ya que si no se hace se corre un gran riesgo de perder beneficios si el crecimiento de la demanda es alto.

Otro efecto que se observa es la convergencia de los resultados cuando la malla de elementos finitos se refina cada vez más; al duplicar el número de puntos, el porcentaje obtenido se hace mayor, pero en general la diferencia entre los resultados obtenidos en dos niveles de refinamiento consecutivos son menores cuanto mayores sean los niveles.

Por último, es digno de mención el hecho de que la mayoría de valores para los cuales se deben añadir dos portadores por estación base superan en 100 % de ocupación del ancho de banda. Es decir, el aumentar el ancho de banda en dos niveles no es rentable a no ser que la demanda supere (en algunos casos) el ancho de banda máximo del sistema; esto implica que los usuarios van a experimentar muchos problemas a la hora de utilizar el sistema, ya que no habrá suficiente ancho de banda para satisfacer sus peticiones. Así, un usuario que intente realizar llamadas observará que hay un alto grado de bloqueo en el sistema,

Paso de tiempo (Δt)	$\frac{1}{250}$
Paso de ancho de banda (h)	$\frac{1}{70}$

Cuadro 7.10: Datos numéricos usados

Portadores añadidos	Refinamiento	Mensual	Trimestral	Semestral	Anual
1	0	77.62 %	63.95 %	55.57 %	40.57 %
	1	80.51 %	68.69 %	59.90 %	44.14 %
	2	80.75 %	71.16 %	61.09 %	47.89 %
	3	81.50 %	71.24 %	62.23 %	48.87 %
2	0	139.07 %	126.57 %	120.55 %	97.91 %
	1	143.68 %	130.84 %	124.65 %	98.64 %
	2	146.05 %	133.04 %	128.31 %	101.77 %
	3	146.41 %	132.56 %	128.61 %	103.37 %

Cuadro 7.11: Resultados para los diferentes niveles de refinamiento de malla

	Añadir un portador			Añadir dos portadores		
	$k=0.03$	$k=0.1$	$k=0.17$	$k=0.03$	$k=0.1$	$k=0.17$
Mensual	81.50 %	84.64 %	87.85 %	146.05 %	152.32 %	158.34 %
Trimestral	71.25 %	74.19 %	77.19 %	132.56 %	139.00 %	144.75 %
Semestral	62.23 %	64.98 %	68.36 %	128.61 %	134.95 %	140.62 %
Anual	48.87 %	51.81 %	54.84 %	103.37 %	109.78 %	116.39 %

Cuadro 7.12: Resultados para diferentes valores de k

y por lo tanto tendrá que realizar varios (posiblemente muchos, si el grado de demanda está alrededor del 150 % de la capacidad máxima de la red) intentos antes de poder conseguir hacer su llamada. Dada la poca calidad del servicio, seguramente el número de clientes se reducirá, debido al movimiento a otros operadores, reduciéndose la demanda de ancho de banda; por tanto, será difícil que se alcancen los valores necesarios para realizar la mejora. Sin embargo, este efecto se puede reducir si se incluye en el modelo el ya mencionado factor de seguridad, que penaliza las situaciones en las que existe mucho bloqueo en el sistema. Los resultados de aplicar esta penalización se muestran más adelante.

En la tabla 7.12 se muestra el efecto del parámetro k en la solución. En ella se muestran los resultados para los intervalos de decisión usados en la tabla anterior (mensual, trimestral, semestral y anual) y para tres valores de k : 0,03, 0,1 y 0,17. El parámetro k es especialmente delicado debido a que, a diferencia de dicho parámetro en el contexto de las redes de acceso a Internet, en las redes de telefonía móvil resulta de difícil estimación. Existen varios métodos para aproximar su valor real, pero los diferentes valores que proporcionan no son próximos entre sí; por ello, en esta tabla se muestra la variación que aparece en los resultados cuando se utilizan diferentes valores de dicho parámetro, lo que permite valorar la influencia de k y por tanto de los métodos utilizados para calcularlo. Se observa que el parámetro k no tiene una

Factor de seguridad	Añadir un portador	Añadir dos portadores
NO	64.98 %	134.95 %
100 %	55.34 %	114.16 %
90 %	52.81 %	107.62 %
80 %	50.33 %	101.27 %

Cuadro 7.13: Influencia del factor de seguridad Φ

gran influencia en las decisiones a tomar, así que su estimación no tiene por qué ser muy precisa para obtenerse resultados válidos.

La siguiente tabla de resultados (7.13) muestra el efecto del uso de un factor de seguridad para penalizar las ya mencionadas situaciones en las cuales la demanda supera el ancho de banda máximo del sistema, mediante una reducción de los beneficios obtenidos (que modela la pérdida de clientes que eligen cambiar de operador). Para ello, se muestra el efecto de introducir dicho factor con diferentes valores en las decisiones tomadas semestralmente con un $k = 0,1$. Los valores de la primera fila muestran el porcentaje de utilización para la mejora cuando no existe el factor de seguridad (estos valores son idénticos a los mostrados en las celdas correspondientes de la tabla 7.12). Cuando se introduce el factor de seguridad, se observa que el porcentaje de utilización mínimo para realizar la mejora del clúster disminuye a medida que disminuye el valor del factor de seguridad; esto es intuitivo, ya que al disminuir el factor Φ lo que se hace es reducir el beneficio en las zonas de mayor ancho de banda, de forma que resulta más rentable mejorar el clúster cuando la demanda es menor. Si no se hiciese así, las pérdidas por la calidad de servicio harían que el beneficio percibido por la empresa fuese menor que si se aumentase el ancho de banda del clúster, ya que en este caso no existiría la fuga de clientes, sino que probablemente la demanda siguiese aumentando. Por ello, es más realista el uso del factor de seguridad Φ a la hora de realizar la toma de decisiones en redes de telefonía móvil.

7.4. Resultados del cálculo distribuido

Los resultados que se muestran en esta sección se refieren al área del sistema de cálculo distribuido implementado. Estos resultados se dividen en dos partes, los resultados de las pruebas del sistema por separado y los resultados del sistema aplicado a los problemas de valoración de redes de fibra óptica y telefonía móvil.

7.4.1. Resultados del sistema de paralelización

Para la prueba del sistema de cálculo distribuido se ha implementado la resolución de un problema tipo, la búsqueda de números primos; este problema también se utiliza como base para calcular la calidad de los nodos (ver sección 6.4.2). El problema consiste en la búsqueda de todos los números primos entre 0 y una cota máxima *max*. El código que se ha utilizado para realizar esta búsqueda es deliberadamente poco eficiente. El pseudocódigo del algoritmo es el siguiente:

```

para cada número impar k mayor que 0 y menor que max {
    para cada número impar j menor que k y mayor que 1 {
        si k es divisible por j
            entonces k no es primo
    }
    si no se ha determinado que k no es primo
        entonces k es primo
}

```

El resultado final es la lista de todos los números primos entre 0 y *max*. A la hora de paralelizar la resolución del problema, se considera que el espacio de búsqueda $[0, max]$ puede ser dividido en subintervalos o bloques, buscándose los números primos incluidos en el bloque de forma independiente de la búsqueda en otros bloques. Así, se puede enviar a nodos distintos peticiones de búsqueda en bloques distintos para que realicen dichas búsquedas en paralelo. Los resultados que devuelvan, una lista de números primos por cada uno de los nodos implicados, serán mezclados en una sola lista por el programa cliente que resuelve el problema completo. A la hora de realizar pruebas con este problema de forma paralela se tendrán que tener en cuenta dos factores adicionales, el tamaño del bloque (t_b) y el número de nodos de la red; estos dos factores se suman al valor *max*, obteniéndose un problema con tres factores relevantes a la hora de analizar la ejecución del código.

El problema de los números primos tiene una característica que es relevante a la hora de su análisis de tiempos de ejecución: el tamaño de la solución crece cuanto más grande se toma *max*, ya que en un intervalo *A* contenido en un intervalo *B* habrá como mucho el mismo número de números primos que en *B*, y posiblemente habrá menos. Igualmente sucede con las soluciones parciales y el tamaño de bloque, cuanto más grandes se tomen los bloques más números primos se encontrarán en cada bloque. Esto hace que el transmitir los resultados consuma un tiempo dependiente del tamaño de bloque que se tome, lo cual repercute negativamente en la mejora de rendimiento obtenida al ejecutar el código distribuido. Así, cuando se aumenta el tamaño de bloque hay dos tendencias opuestas:

- Dado que el tamaño de bloque es mayor, la búsqueda será más larga, por lo que el tiempo que consumirá la tarea será mayor. Este hecho es beneficioso, ya que el aumentar el tiempo de ejecución

<i>max</i>	500000	1000000	5000000	10000000
Tiempo (ms)	4806	12305	111380	295511

Cuadro 7.14: Tiempos de ejecución secuencial para distintos valores de *max*

de la tarea con respecto al tiempo de comunicación hace que mejore el rendimiento global del sistema (como ya se comentó en la sección de implementación, el sistema de cálculo distribuido se comportará mejor cuanto más grandes sean las tareas a resolver)².

- Por otra parte, al aumentar el tamaño de bloque se aumenta el número de primos en el bloque, por lo que se transmitirá un resultado mayor cuando la tarea se complete. Esto hace que aumente el tiempo de comunicación, lo que perjudica a la mejora de velocidad del código paralelo con respecto al código secuencial. Efectivamente, para una duración de tareas fija, el sistema se comportará mejor cuanto menor sea el tiempo de comunicación.

Si el problema tuviese una solución de tamaño fijo, entonces el único efecto que aparecería sería el primero, con lo que el aumentar el tamaño de los bloques siempre mejoraría el resultado, ya que las tareas serían más largas y el tiempo de comunicación sería el mismo; sin embargo, la mejora de rendimiento al utilizar el código paralelo es menor que la esperada si el resultado tiene un tamaño variable, ya que aparece el segundo factor que disminuye la velocidad de comunicación de los resultados de las tareas.

La medida que se utilizará en las pruebas será el tiempo de ejecución del problema, en concreto la razón entre el tiempo de ejecución del problema secuencial y el tiempo de ejecución del problema distribuido; los resultados de ambos programas deben ser los mismos, de forma que se considerará esta situación como normal, y solamente se realizarán comentarios en este sentido en el caso de que los resultados del programa paralelo difiriesen de los obtenidos mediante el programa secuencial.

Los resultados se han obtenido ejecutando el código en ordenadores con procesador Intel PentiumIII 600Mhz, con 128 megabytes de memoria RAM, conectados mediante una red Ethernet 10/100.

Ejecución secuencial

En un principio se ejecuta el código secuencial con varios valores de *max*, obteniéndose los tiempos de ejecución mostrados en la tabla 7.4.1 y en la Figura 7.5.

²Se está considerando que el número de tareas es constante, independientemente del tamaño de bloque, lo que implica que al aumentar el tamaño de bloque se aumenta el valor de *max*. Si el valor de *max* no se actualiza, entonces al aumentar el tamaño del bloque se mejora el beneficio por tarea a ejecutar, pero se disminuye el número de tareas y por tanto la capacidad de paralelismo aprovechada. En el límite se comprueba que al hacer el tamaño de bloque igual a *max* el beneficio obtenido por tarea es máximo, ya que la tarea será lo más larga posible, pero solamente se estará ejecutando una búsqueda, con lo que el problema se estará resolviendo de forma secuencial (de hecho, el rendimiento será más pobre, ya que hay que sumar los tiempos de comunicación y preparación del sistema al tiempo de cálculo secuencial)



Figura 7.5: Tiempos de ejecución secuencial para distintos valores de max

Como era de esperar, el tiempo de ejecución aumenta con el valor de max .

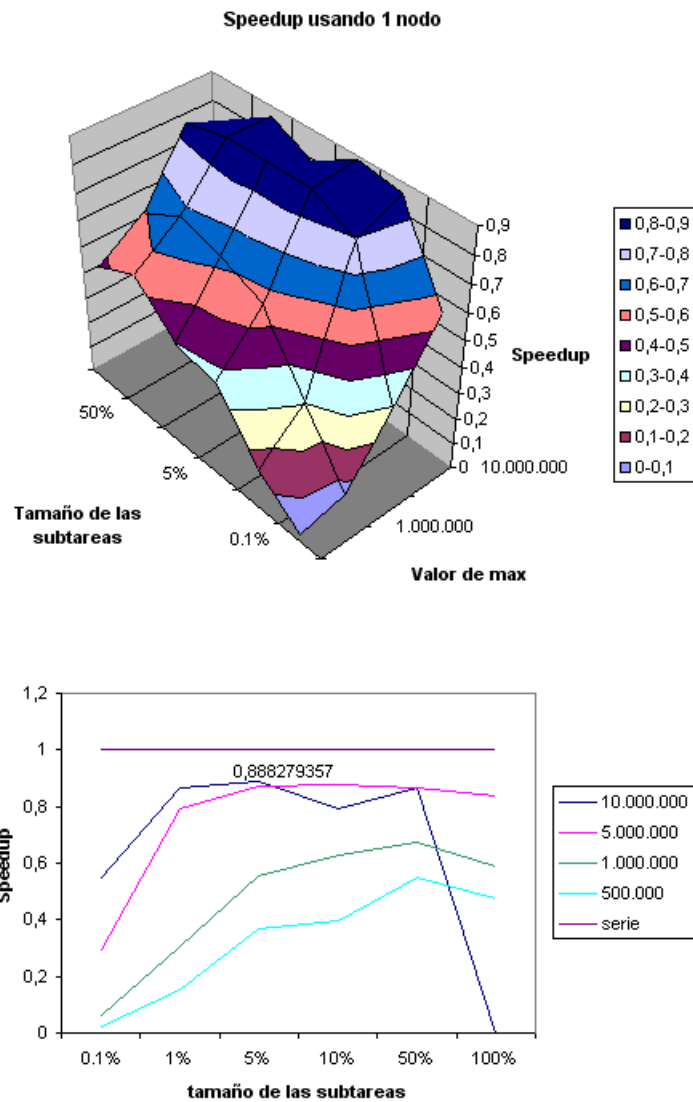
Ejecución distribuída

Para comparar los tiempos de ejecución, se prueba el código paralelo para cada uno de los valores de max utilizados en la ejecución secuencial. Para cada uno de estos valores, se probarán diversas configuraciones de la red (modificando el número de nodos disponibles) y varios valores para el tamaño de bloque. En concreto, se tomarán los tamaños de bloque que representen el 0,1 %, 1 %, 5 %, 10 %, 50 % y 100 % del valor de max . Esto significa que el problema se subdividirá en 1000, 100, 20, 10, 2 y 1 tareas, dependiendo de la ejecución que se realice. ³

Los resultados no se presentarán en valor absoluto de milisegundos, sino que se calcula el *speedup* como $\frac{t_{serie}}{t_{paralelo}}$; así, la ejecución en un ordenador tendrá un *speedup* de 1. Valores menores que este representarán tiempos de ejecución mayores, mientras que valores mayores representarán tiempos de ejecución menores que los tiempos de ejecución de un ordenador. El objetivo del sistema de cálculo distribuído es obtener el mayor *speedup* posible.

- Resultados para 1 nodo (tabla 7.15 y Figura 7.6): como es de esperar, los resultados obtenidos utilizando un nodo remoto siempre serán peores que la ejecución del problema en un ordenador, ya que el tiempo empleado será el utilizado por un ordenador para la resolución del problema más el tiempo necesario para la transmisión de las especificaciones y resultados a través de la red. En efecto, los valores de *speedup* conseguidos son los mostrados en la tabla 7.15, En esa tabla se presenta el

³el *speedup* para $max = 10000000$ y $t_b = 100\%$ no se ha podido calcular con el hardware disponible, por falta de memoria RAM

Figura 7.6: *Speedup* obtenido utilizando un nodo

speedup para cada uno de los valores de *max* utilizados (500000,1000000,5000000 y 10000000), y para cada tamaño de bloque elegido (0,1 %,1 %,5 %,20 %,50 % y 100 %).

Obviamente, es preferible resolver el problema con un ordenador de forma local a utilizar el sistema de cálculo distribuído si solamente va a existir un nodo de cálculo.

- Resultados para 2 nodos (Figura 7.7): al añadir un segundo nodo, se comienza a explotar de forma efectiva las capacidades de paralelización del problema, como se observa en la tabla 7.16. En efecto, se puede observar que excepto en el caso de la búsqueda entre 0 y 500000, el uso de dos nodos hace que los tiempos de cálculo sean menores que en la ejecución en serie (el *speedup* es mayor). Como se había previsto, el *speedup* es mayor cuanto mayor sea el valor de *max*, manteniendo el porcentaje

Tamaño subtareas	10000000	5000000	1000000	500000
0.1 %	0,54	0,29	0,06	0,02
1 %	0,86	0,79	0,30	0,15
5 %	0,88	0,86	0,55	0,36
10 %	0,79	0,87	0,62	0,39
50 %	0,86	0,86	0,67	0,55
100 %	-	0,83	0,58	0,47

Cuadro 7.15: Speedup obtenido utilizando un nodo de cálculo

Tamaño subtareas	10000000	5000000	1000000	500000
0.1 %	0,66	0,39	0,13	0,03
1 %	1,68	0,76	0,64	0,20
5 %	1,68	0,91	1,23	0,52
10 %	1,65	1,66	1,17	0,67
50 %	0,90	1,35	1,13	0,69
100 %	-	0,83	0,63	0,46

Cuadro 7.16: Speedup obtenido utilizando dos nodos de cálculo

de *max* que representa el tamaño de cada bloque.

Así, el mayor *speedup* obtenido es de 1,69 aproximadamente, que representa un valor muy razonable, teniendo en cuenta que la ejecución se realiza en un sistema real, no ideal, en el que existen las penalizaciones ya comentadas.

- Resultados para 4 nodos (tabla 7.17 y Figura 7.8): al añadir dos nodos de cálculo más se produce un incremento en el rendimiento, aumentando el *speedup* máximo a 3,16 aproximadamente. Este rendimiento máximo se obtiene, como era de esperar, en la búsqueda más larga (entre 0 y 10000000). En general, los resultados mejores se consiguen con un tamaño de bloque intermedio, entre el 1 % y el 5 % (es decir, dividiendo el problema en un número de subtareas comprendido entre 20 y 100). Con un tamaño menor, el peso de las comunicaciones se hace patente y degrada el rendimiento en gran medida, mientras que si se divide el problema en dos o incluso una sola subtarea no se explota el paralelismo hardware que existe, ya que se desaprovechan dos o tres nodos, según el caso.
- Por último, se presentan los gráficos del *speedup* para la resolución con 6 nodos de cálculo (Figura 7.9), que representa el conjunto de más capacidad computacional utilizado. En este caso el rendimiento sigue aumentando, llegándose a un *speedup* máximo de aproximadamente 4,27 (ver

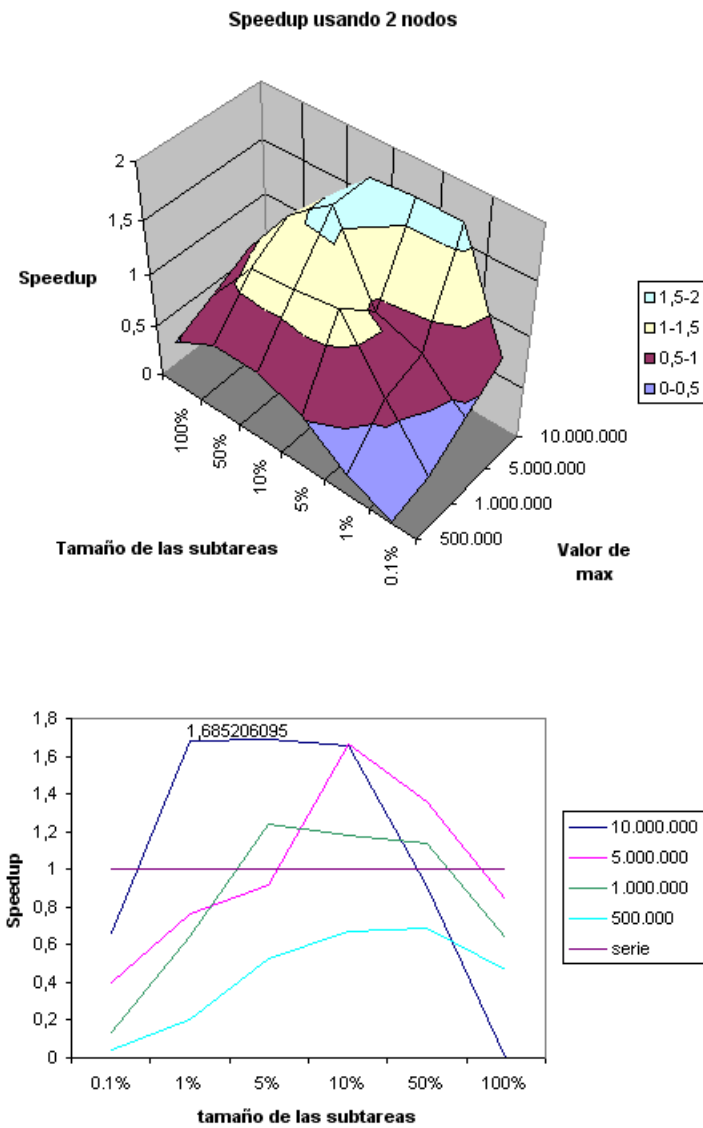


Figura 7.7: *Speedup* obtenido utilizando dos nodos

Tamaño subtareas	10000000	5000000	1000000	500000
0.1 %	1,92	1,29	0,17	0,03
1 %	3,16	2,71	0,98	0,24
5 %	3,09	2,93	1,54	0,65
10 %	2,78	2,57	1,49	0,78
50 %	1,37	1,32	1,11	0,69
100 %	-	0,85	0,69	0,47

Cuadro 7.17: *Speedup* obtenido utilizando cuatro nodos de cálculo

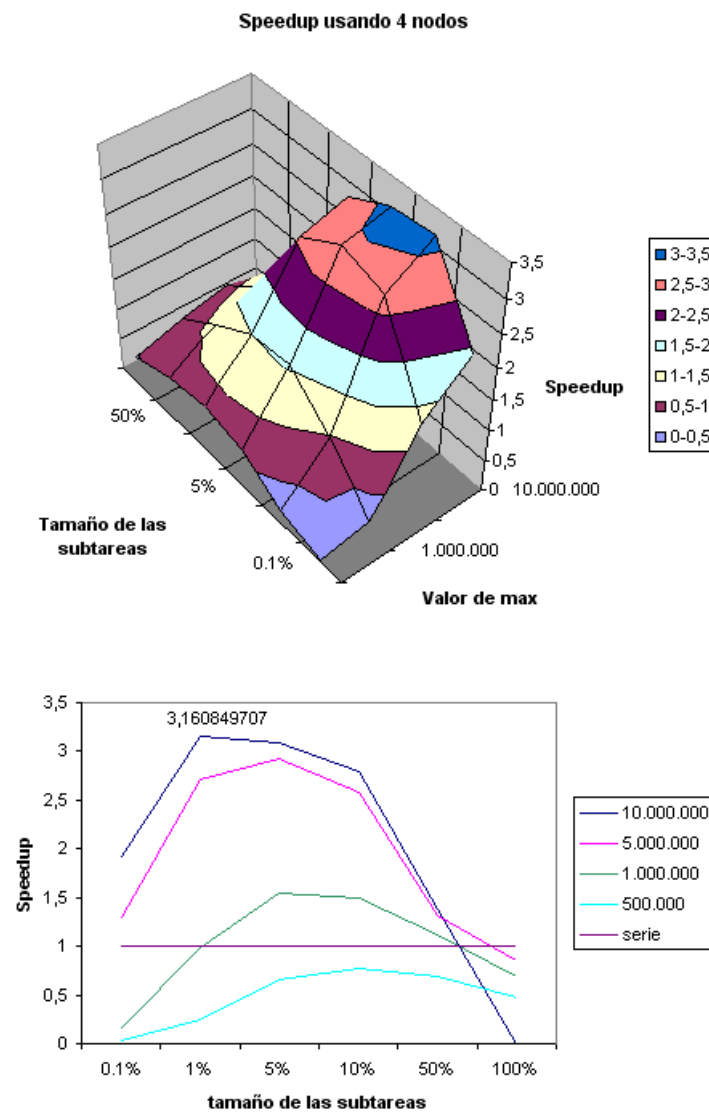


Figura 7.8: *Speedup* obtenido utilizando cuatro nodos

Tamaño subtareas	10000000	5000000	1000000	500000
0.1 %	2,35	1,51	0,23	0,10
1 %	4,22	3,46	1,06	0,57
5 %	4,27	3,73	1,58	1,00
10 %	3,87	3,31	1,64	1,01
50 %	1,40	1,37	1,15	0,75
100 %	-	0,85	0,70	0,55

Cuadro 7.18: Speedup obtenido utilizando seis nodos de cálculo

tabla 7.18). Los efectos observados en el caso de ejecuciones con 4 nodos se mantienen cuando se usan 6 nodos: los mejores resultados se obtienen con tamaños de subtareas intermedios y para valores de *max* altos; cuanto más se alejan los parámetros de esos valores óptimos, peor rendimiento resulta.

En las figuras 7.10 y 7.11 se muestra la comparación del *speedup* obtenido para la ejecución del mejor y el peor caso del valor de *max* (10000000 y 500000 respectivamente), para los diferentes nodos utilizados.

Por último, en la Figura 7.12 se observa la evolución del *speedup* máximo según varía el número de nodos utilizados. Se aprecia que el aumento de rendimiento es lineal en el número de nodos utilizados, aunque es previsible que cuando el número de nodos de la red aumenta mucho el rendimiento sea peor, debido a que el nodo central debe atender peticiones más rápidamente de las que posiblemente pueda atender, y a que el programa cliente recibirá resultados a mayor velocidad de la que es capaz de alcanzar integrando dichos resultados parciales en la solución total.

Así, con estos resultados se comprueba que el rendimiento utilizando el sistema de cálculo distribuído aumenta de forma satisfactoria cuando se utilizan varios ordenadores de propósito general, unidos con una red de comunicaciones no especializada. Además, la programación de aplicaciones con este sistema es extremadamente sencilla, por lo que los objetivos marcados a la hora de construir el sistema de cálculo distribuído se consideran completados.

7.4.2. Valoración de redes con cálculo distribuído

En esta sección se muestran los resultados de aplicar el sistema de paralelización desarrollado a los problemas objetivo del presente Proyecto Fin de Carrera. A diferencia del problema de búsqueda de números primos, que fue elegido como problema de prueba, en la toma de decisiones en redes de acceso a Internet y de telefonía móvil no se pueden ajustar todos los parámetros del problema para optimizar la resolución en paralelo. En efecto, en la búsqueda de números primos se puede modificar el tamaño de bloque (el intervalo en el que busca cada una de las tareas). Esto repercute directamente en el tiempo de

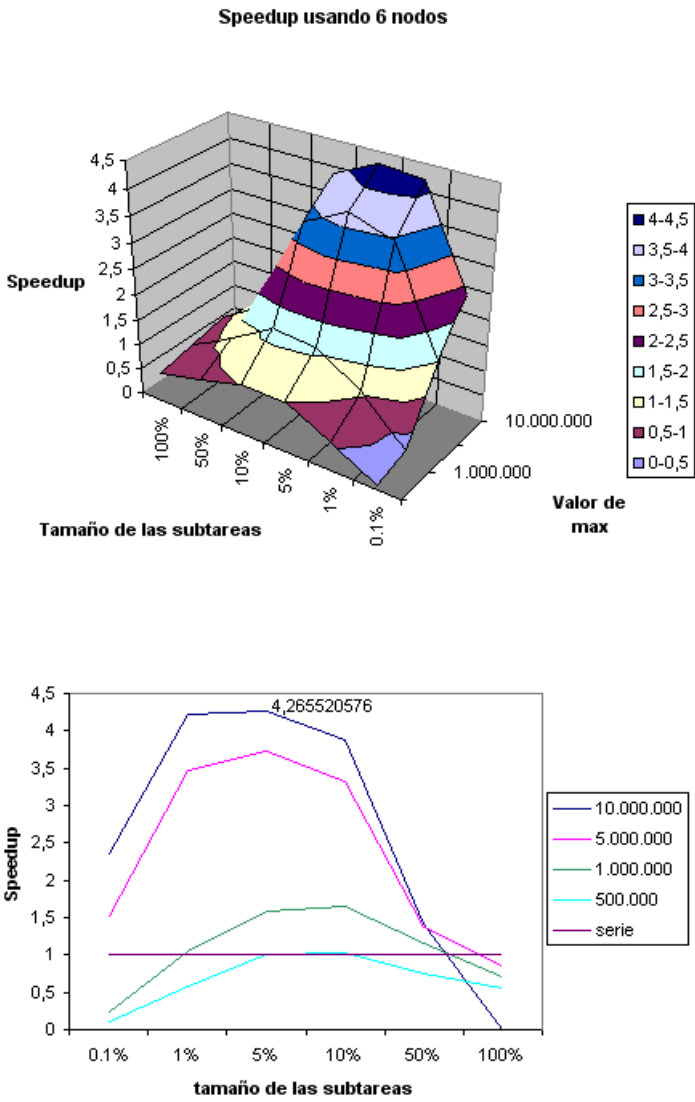


Figura 7.9: *Speedup* obtenido utilizando seis nodos

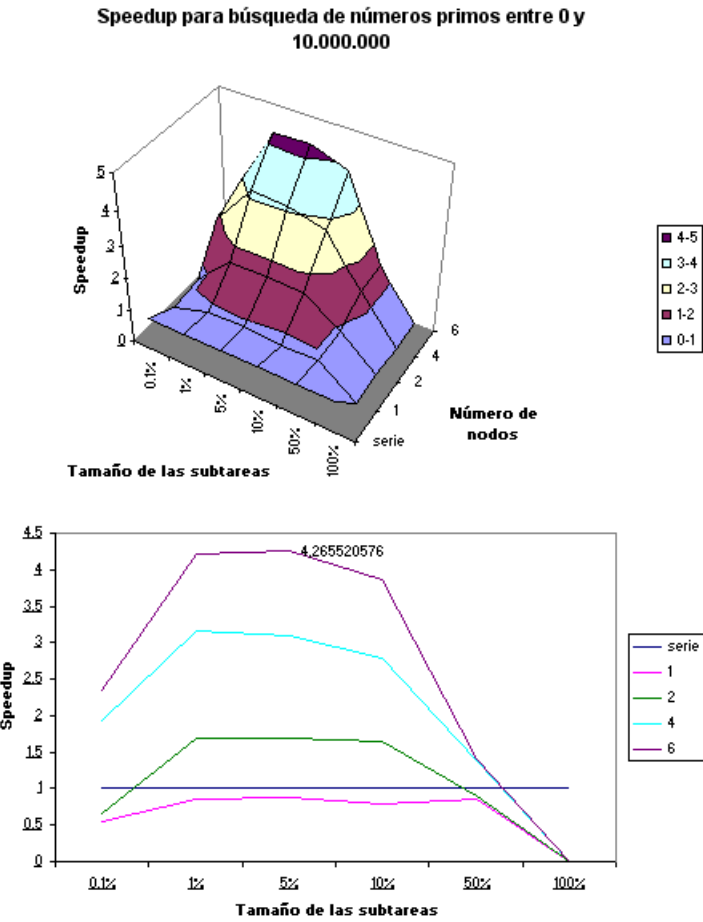


Figura 7.10: *Speedup* para el mejor valor de *max* (10000000)

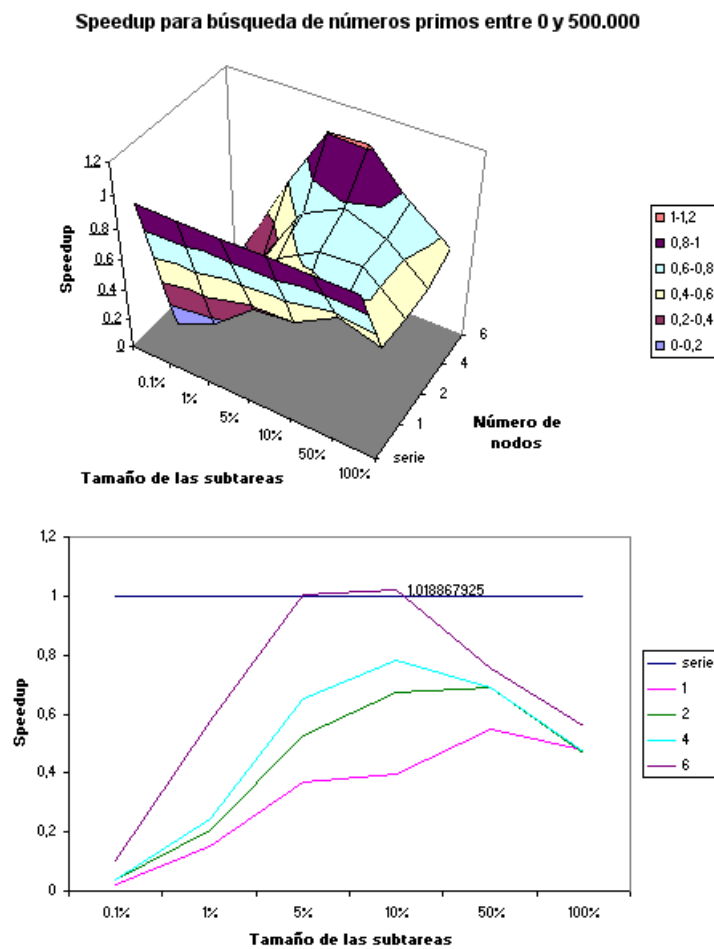


Figura 7.11: *Speedup* para el peor valor de *max* (500000)



Figura 7.12: Evolución del mejor *speedup* según el número de nodos de cálculo

resolución, como se ha explicado en la sección anterior, dado que determina tanto el número de tareas que existen como la duración de cada una; dependiendo del número de nodos disponibles en la red, y de la calidad de las comunicaciones en la red que los une, se puede ajustar ese parámetro para obtener resultados óptimos.

Sin embargo, en la valoración de redes el número de subtareas está fijado por la estructura del problema. Recuérdese que en estos dos problemas cada subtaska está encargada de calcular el valor de una línea o clúster (dependiendo de qué problema se resuelva) entre dos fechas relevantes del problema (ver capítulo 5). Así, el número de subtareas no depende de ningún factor independiente que se pueda ajustar al definir el problema, sino que viene determinado por varios factores: la zona en la que se esté resolviendo el problema en un momento dado (las zonas de tipo I tienen menos subtareas que las zonas de tipo II), el número de líneas del problema, la distribución de las fechas relevantes... Estos factores no pueden ser ajustados libremente, ya que vienen determinados por la situación real que se quiere modelar; por ello, esos valores vienen determinados por el entorno financiero y empresarial, y, por tanto, son fijos (o permiten unos ajustes relativamente pequeños, y en ningún caso con el objetivo de mejorar el cálculo paralelo). El único factor no determinado por el contexto del problema es el relacionado con el método de elementos finitos: la densidad de la malla utilizada. Este es un factor artificial (en el sentido que no viene determinado por el problema, sino que se introduce debido a la forma de resolverlo), por lo que puede ser ajustado libremente. Dicha densidad de la malla (especificada como el número de puntos en ancho de banda y el paso de tiempo) determinará la duración de cada tarea, sin afectar al número de las mismas; claramente, la densidad de la malla afecta al tiempo de resolución del método de elementos finitos. Sin embargo, la densidad de la malla no puede elegirse libremente, ya que determina la precisión con la que se resuelve el problema: mallas menos densas harán que el problema se resuelva mucho más rápidamente que mallas con un mayor número de puntos, pero los resultados que se obtengan con ellas serán menos fiables. De esta forma, la densidad de malla siempre debería tomarse como la máxima densidad que soporte el hardware disponible y que haga que el problema se resuelva dentro de un tiempo asumible. En este sentido, la paralelización puede ayudar a obtener más precisión en los resultados, al poder utilizar mallas más finas resolviendo el problema en aproximadamente el mismo tiempo que un solo ordenador con una malla con menos puntos.

Sin embargo, la aplicación del cálculo distribuido tiene una penalización adicional en el caso de los problemas planteados. En efecto, en la sección 6.1.2 se mostraron las optimizaciones aplicadas al método de elementos finitos con características en su implementación. La principal de estas mejoras residía en calcular una única vez la matriz del sistema A_h , de forma que esa matriz se almacenaba y se reutilizaba a lo largo de todo el código. Sin embargo, el sistema de cálculo distribuido impone a las clases que resuelven los problemas la restricción de que no pueden guardar ninguna clase de estado dependiente del cliente que reserve el nodo; así, no se puede guardar la matriz del sistema de un cliente, ya que otro cliente

$h = \Delta t$	Serie	2 Nodos	4 Nodos	6 Nodos
0.01	1232	15060	18704	18695
0.005	5307	17854	19796	20808
0.0025	17203	23641	22049	24602
0.001	98860	68270	45149	44359
0.0005	400528	219760	143950	106160
0.00025	1672495	852263	514298	425233

Cuadro 7.19: Tiempos obtenidos en el problema de toma de decisiones para redes de acceso a Internet

podría querer resolver otra ecuación diferente, además de plantearse problemas como cuándo eliminar la matriz del sistema (un nodo no tiene por qué saber que se ha terminado la resolución del problema). Así, la optimización aplicada solamente es válida parcialmente: cada nodo ha de recalcular la matriz A_h cada vez que inicia la resolución de una subtarea. Esa matriz la almacena y la utiliza durante la resolución de toda la subtarea, pero, al terminar ésta, se descarta. Este efecto provoca que la mejora en el tiempo de ejecución que se obtendrá en los problemas de toma de decisiones será menor que en el caso de la búsqueda de números primos, debido a la imposibilidad de aplicar todas las optimizaciones aplicadas en el caso de resolución en un único procesador.

Por último, antes de presentar los resultados obtenidos, se ha de comentar el hecho de que los resultados obtenidos cuando se resuelven de forma distribuida los problemas presentados son exactamente iguales que los obtenidos al resolver dichos problemas de forma secuencial. Esto es algo vital, ya que el cálculo distribuido no debe alterar a las soluciones calculadas, sino solamente a la velocidad con la que se calculan esas soluciones.

Toma de decisiones para redes de acceso a Internet

A continuación se muestran los resultados referidos a la paralelización en el problema de las redes de fibra óptica. Para obtener dichos resultados se ha utilizado la misma infraestructura hardware que en el caso del problema de búsqueda de números primos.

En la tabla 7.19 se muestran los tiempos de cálculo obtenidos (en milisegundos), y en la Figura 7.13 se muestran los diferentes *speedup* obtenidos, dependiendo de la precisión utilizada y del número de nodos en la red. Como era de esperar, cuando se utiliza una malla muy basta los tiempos obtenidos con la resolución distribuida son mucho peores que los obtenidos con un solo ordenador; en estos casos, la sobrecarga que imponen las comunicaciones y la imposibilidad de aplicar las optimizaciones de forma completa causan este deterioro del rendimiento. Sin embargo, cuando se utilizan mallas más finas, el *speedup* obtenido crece hasta situarse en un valor de aproximadamente 3,9 cuando se utilizan 6 nodos de cálculo ⁴. De esta

⁴En realidad, los tiempos tomados con 6 nodos están ligeramente sesgados, ya que el sexto nodo de cálculo estaba situado

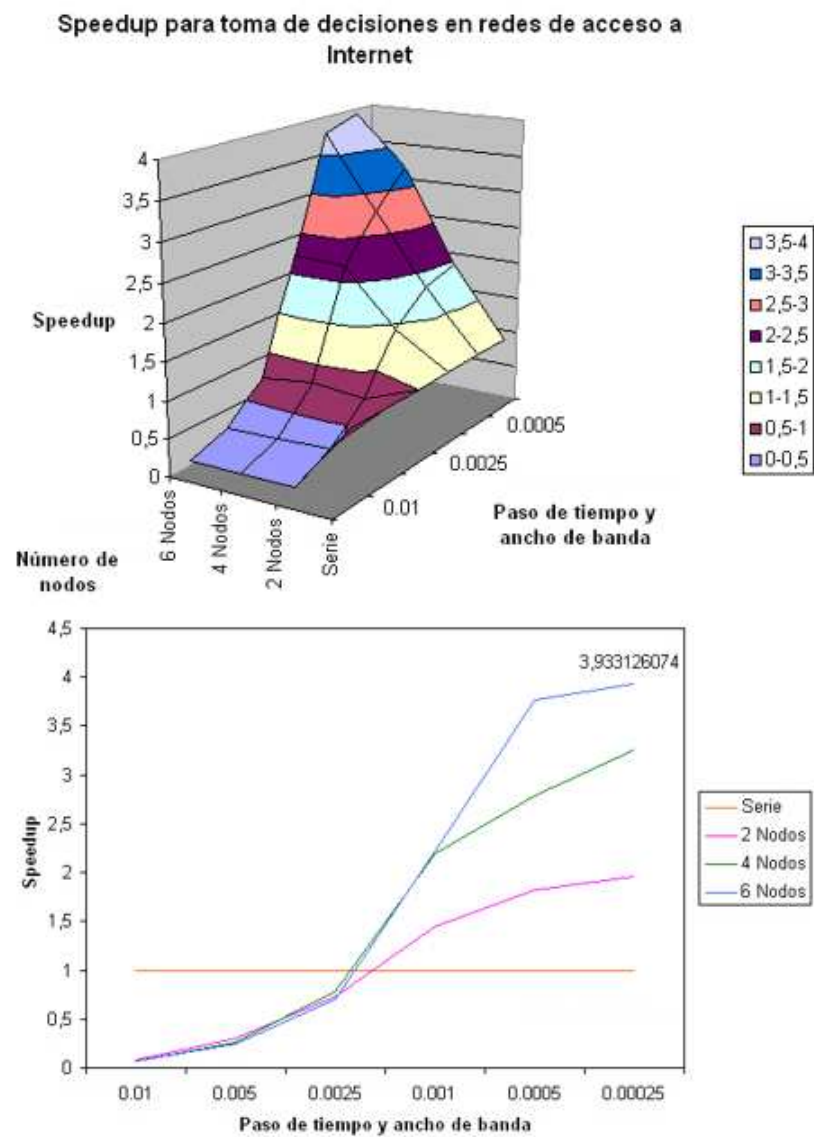


Figura 7.13: *Speedup* obtenido en el problema de toma de decisiones para redes de acceso a Internet

forma, cuanto más fina es la malla de elementos finitos que se toma, más aumenta el *speedup* obtenido, debido a que las comunicaciones representan una parte menor del problema frente a la resolución de las ecuaciones en los nodos (que al ser la malla más fina requieren mucho más tiempo).

Toma de decisiones para redes de telefonía móvil

Por último, se presentan los resultados de las redes de telefonía móvil. La configuración hardware utilizada es la misma que en la valoración de redes de fibra óptica. La diferencia entre ambos problemas referente al tiempo de cálculo es la mayor complejidad del problema de telefonía móvil. Como ya se mostró en el capítulo 3, la posibilidad de solaparse varias decisiones y la capacidad de fraccionar los pagos se traducen en un aumento considerable del número de tareas que se han de resolver entre cada dos fechas significativas con respecto al problema de redes de acceso a Internet. Así, es de esperar que tanto los tiempos de ejecución como la memoria utilizados en la resolución de dicho problema sean mayores; sin embargo, la complejidad de cada tarea es muy similar (recuérdese que en el caso de la telefonía móvil el pago de los plazos de mantenimiento no está incluido entre las responsabilidades de cada tarea, mientras que en el caso de las redes de acceso a Internet sí). Pese a esto, es previsible que la mejora en el tiempo de ejecución sea menor en el caso de la valoración de redes de telefonía móvil que en las redes de acceso a Internet, debido a que las tareas son en general más pequeñas. En el caso de las redes de fibra óptica, cada tarea resolvía la ecuación de una línea entre una fecha de notificación y una de mejora (o viceversa). En el caso de las redes de telefonía móvil, cada una de las tareas resuelve la ecuación de un clúster entre dos fechas de observación consecutivas. En general, las fechas de observación están mucho más próximas entre sí que las fechas relevantes en el problema de las redes de acceso a Internet, por lo que las tareas en el problema de telefonía móvil serán mucho más cortas, haciendo que el efecto de las comunicaciones entre nodos aumente considerablemente con respecto al caso de las redes de fibra óptica, por lo que el *speedup* esperado es menor.

En la tabla 7.20 se muestran los tiempos obtenidos para diferentes mallas y usando diferente número de nodos; en la Figura 7.14 se pueden observar los *speedup* obtenidos en cada caso. Es importante señalar que la configuración con 6 nodos de los problemas anteriores ha sido sustituida por una con 5 nodos, ya que debido a las limitaciones hardware en el momento de las mediciones, surgía el mismo problema que en el caso de las redes de fibra óptica (ver la nota al pie en la página 168). Sin embargo, debido a que el problema de las redes de telefonía móvil ejecuta muchas más subtareas que el de las redes de acceso a Internet, en este caso la degradación en el rendimiento provocada por ejecutar en el mismo nodo

físicamente en el mismo ordenador en el que se ejecutaba la aplicación cliente, el nodo central y el servicio de nombres, por lo que se reducía la eficiencia de las comunicaciones, del mecanismo de reserva y liberación de nodos y del cálculo por parte del sexto nodo. Así, en un entorno con 7 o más ordenadores, la ejecución con 6 nodos de cálculo necesariamente tendrá que presentar mejores tiempos que los mostrados aquí. Los obtenidos en el presente Proyecto Fin de Carrera fueron tomados solamente con 6 ordenadores debido a problemas de disponibilidad del hardware

$h = \Delta t$	Serie	2 Nodos	4 Nodos	6 Nodos
0.01	992	85012	91100	89948
0.005	4426	86604	73617	71774
0.0025	15361	98110	77672	78945
0.001	89170	167200	111207	103346
0.0005	391623	355689	207323	187787
0.00025	1673147	1212358	666508	504411

Cuadro 7.20: Tiempos obtenidos en el problema de toma de decisiones para redes de telefonía móvil

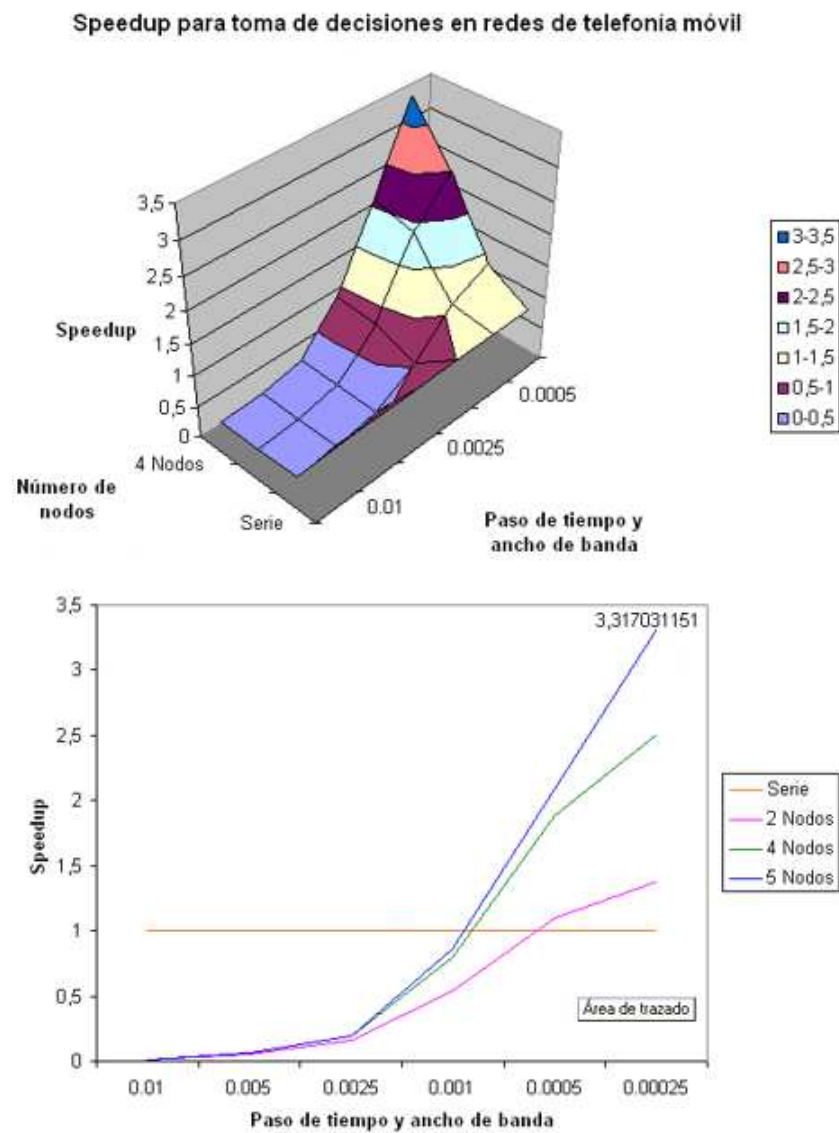


Figura 7.14: *Speedup* obtenido en el problema de toma de decisiones para redes de telefonía móvil

el servicio de nombres, la aplicación cliente, el nodo central y el nodo de cálculo era tan grande que se optó por eliminar el nodo de cálculo y medir los tiempos solamente para 5 nodos. Este problema se soluciona utilizando más ordenadores, u ordenadores con mayor capacidad computacional (los usados en la obtención de estos resultados son relativamente antiguos).

Los efectos observados son idénticos al caso de las redes de fibra óptica: al utilizar mallas poco finas, los tiempos obtenidos son mucho peores si se utiliza el cálculo distribuido. Sin embargo, cuando las mallas se toman finas, se hace patente la ventaja de utilizar varios ordenadores en la resolución del problema, consiguiendo un tiempo de resolución menor de un tercio del original, en el mejor de los casos. Dos apuntes son necesarios: en primer lugar, observar que debido al mayor peso de las comunicaciones en este problema, es necesario tomar mallas más finas para que el tiempo se reduzca con respecto a la ejecución en un ordenador. Así, en el caso de la fibra óptica, bastaba con tomar $h = \Delta t = 0,001$ para obtener mejores resultados (con 2 nodos de cálculo, el *speedup* obtenido es de alrededor de 1,44); sin embargo, en el caso de las redes de telefonía móvil es necesario tomar un $h = \Delta t = 0,0005$ para mejorar los tiempos (y aún así solamente se obtiene un *speedup* de aproximadamente 1,10). En segundo lugar, comparar la mejora de tiempos obtenida con 4 nodos en los dos problemas, para el mejor caso: en las redes de acceso a Internet, esta mejora es de 3,25, mientras que en las redes inalámbricas es de 2,51: se observa por tanto el efecto que la disminución del tamaño de las tareas tiene en el tiempo de cálculo distribuido.

Parte IV

Conclusiones

Capítulo 8

Conclusiones y trabajos futuros

8.1. Conclusiones

En el presente Proyecto Fin de Carrera se ha mostrado el proceso de construcción de una aplicación software que resuelve dos problemas surgidos en el ámbito empresarial. Como ya se indicó en la introducción, la toma de decisiones referidas a las inversiones de capital siempre son un problema al que las empresas deben prestar mucha atención; en concreto los proveedores de acceso a Internet se enfrentan al problema de decidir cuándo invertir capital en sus redes de acceso, y a qué nivel deben mejorar dichas redes, y los operadores de telefonía móvil se enfrentan a un problema similar con las redes que utilizan para dar servicio a sus clientes. En ambas ocasiones el factor que es la causa de la mayor parte de la incertidumbre asociada a la decisión es la demanda, o la cantidad de servicio que requieren los clientes. En los dos contextos mencionados, la demanda es altamente volátil, de forma que es muy difícil predecirla, y una estimación incorrecta puede desembocar en unos muy malos resultados en cuanto a beneficios percibidos por la empresa se refiere. En efecto, ambos problemas presentan los mismos riesgos: si la empresa propietaria de la red (bien sea de acceso a Internet o de telefonía móvil) no invierte suficiente capital en su infraestructura, no será capaz de satisfacer toda la demanda presente, y perderá beneficios respecto a la situación ideal, perdiendo clientes (ya que probablemente éstos se cambien a proveedores que ofrezcan un servicio de mayor calidad); sin embargo, si la empresa invierte demasiado capital en sus redes, éstas estarán sobredimensionadas, lo que quiere decir que la demanda que la empresa recibirá no le proporcionará la renta suficiente como para amortizar la infraestructura adquirida, por lo que el beneficio percibido será más pequeño que si se hubiese realizado una inversión más pequeña, o incluso si no se hubiese realizado ninguna. Esta importancia de las decisiones hace que sea importante contar con guías objetivas que puedan ayudar a tomar las decisiones a los responsables de éstas, evitando los problemas tanto de redes con demasiada capacidad como los de redes con capacidad insuficiente.

Por este motivo, se ha construido una aplicación que proporciona una ayuda a la hora de decidir la

cuantía y el momento de las decisiones que las empresas mencionadas deben afrontar. Esta aplicación muestra las decisiones a tomar según la demanda que la empresa percibe, en cada una de las fechas en las que se deben tomar decisiones de inversión (es una práctica común que estas fechas estén distribuidas periódicamente).

Para construir la aplicación, se ha partido de dos modelos matemáticos basados en la demanda estocástica, uno para cada tipo de redes consideradas. En estos modelos se han completado las áreas que no estaban especificadas en su totalidad. Estos modelos se basan en técnicas de cobertura dinámica para obtener ecuaciones en derivadas parciales de segundo orden que se utiliza para calcular el valor de una línea (en el caso de las redes de acceso a Internet) o clúster (en el de las redes de telefonía móvil) a lo largo del tiempo, en presencia de la incertidumbre en la demanda. Con el valor de las diferentes líneas o clústers se aplican algoritmos de decisión que proporcionan las decisiones óptimas, que maximizan el beneficio obtenido en cada momento.

Para llevar a cabo la resolución de las ecuaciones en derivadas parciales se desarrolla un método de elementos finitos con características, que se muestra válido a la hora de aproximar las soluciones de dichas ecuaciones parabólicas, y al que se aplica una serie de optimizaciones que hacen que se ejecute en un tiempo óptimo.

A partir del método de elementos finitos y de los dos modelos, se ha construido una aplicación software que proporciona al usuario una forma de introducir los datos de entrada de cada uno de los problemas, calcula las decisiones óptimas para cada caso, y muestra dichas decisiones al usuario. Esta aplicación está constuída de forma modular, para que se pueda extender de forma fácil, y con una interfaz gráfica de usuario, que permite que su utilización sea sencilla por parte de usuarios que no están especializados en informática ni métodos numéricos

Con dicha aplicación las empresas pueden calcular las decisiones óptimas para sus decisiones, y utilizarlas como una ayuda a la hora de realizar las inversiones. En efecto, en los resultados mostrados se ha observado que las decisiones óptimas con más conservadoras (en el sentido de que son más reacias a invertir capital) que las prácticas utilizadas en el pasado en el ámbito de la inversión empresarial, consistentes en mejorar las redes cuando se alcance un 50 % de su capacidad; esta práctica conduce, por tanto, a un problema de sobredimensionamiento.

Además, se ha mostrado cómo el lenguaje de programación Java puede ser utilizado para la construcción de aplicaciones que involucren cálculos complejos, siendo lo suficientemente rápidas para que la aplicación se ejecute dentro de límites aplicables en una situación de explotación de la aplicación, desmintiendo la creencia de que dicho lenguaje es demasiado lento para aplicaciones científicas, y permitiendo además reimplementar las partes críticas del programa en lenguajes más rápidos como Fortran o C, en caso necesario.

Por último, se ha realizado un estudio de la viabilidad del uso de cálculo distribuído a la hora de

resolver los problemas planteados. Para aplicar esta técnica, se ha desarrollado un sistema de cálculo distribuido que permite ejecutar tareas de forma remota, que es de alto nivel (extremadamente sencillo de utilizar) y que proporciona resultados positivos a la hora de reducir los tiempos de ejecución de la aplicación. Además, el sistema se ha desarrollado de forma que sea genérico (no específico para esta aplicación) y, por tanto, puede ser utilizado en multitud de aplicaciones sin tener que modificarse.

8.2. Líneas de trabajo futuras

El presente Proyecto Fin de Carrera no debe considerarse un proyecto cerrado, sino que ofrece la posibilidad de ser ampliado en varias áreas, posibilitando la creación de un sistema de mayores prestaciones. Algunas de dichas líneas de trabajo futuras son:

- En el método numérico: implementación de nuevos métodos numéricos para resolver las ecuaciones en derivadas parciales implicadas en los problemas resueltos. A raíz de la inclusión de dichos métodos se pueden realizar comparaciones entre diferentes métodos, comprobando tanto la validez de los resultados como los tiempos de ejecución. Además, el desarrollo de otros métodos numéricos puede resolver los problemas planteados por el método de elementos finitos con características en cuanto a las condiciones de contorno (por ejemplo, los problemas que presenta la condición de contorno en el extremo derecho del dominio de Q).
- En la valoración de inversiones: tanto en la toma de decisiones de redes de acceso a Internet como en la de redes de telefonía móvil se han implementado modelos de un solo factor (la demanda de capacidad). Se puede plantear la implementación de nuevos modelos de más factores para realizar la toma de decisiones de una forma más precisa a priori; por ejemplo, el uso de la demanda de ancho de banda y del precio unitario a la hora de calcular el valor de las redes (modelo de dos factores).

Además, sería natural el desarrollar o implementar nuevos modelos para la valoración de nuevos tipos de redes u otros activos, para la obtención de un conjunto de librerías que resuelvan una gran parte del conjunto de decisiones de inversiones empresariales; así, el paquete software sería de mayor valor para muchas empresas actuales.
- Un punto que puede ser investigado es la posibilidad de aplicar los modelos implementados actualmente a nuevas situaciones, para expandir la aplicabilidad del software. Por ejemplo, se podría estudiar qué modificaciones serían necesarias (en caso de ser posible) para adaptar el modelo de valoración de redes de acceso a Internet a la valoración de cualquier tipo de redes empresariales con cable, quizá mediante el desarrollo de nuevas funciones de beneficio; para la valoración de redes empresariales inalámbricas quizá se podría adaptar el modelo de redes de telefonía móvil, o incluso se podrían unir ambas aproximaciones para realizar un tratamiento completo de las redes internas

de una empresa u organismo.

- En la aplicación software: la aplicación software final ha sido diseñada de forma modular, así que se le pueden realizar inclusiones de nuevos módulos de forma sencilla. Los nuevos modelos que se implementen deberían ser traducidos en nuevos módulos añadidos a la aplicación; además, se puede completar la aplicación con otras herramientas que sean útiles en el entorno empresarial, como, por ejemplo, un generador de informes más elaborado que la actual opción de impresión de resultados, o la posibilidad de la recolección automática de parámetros del problema, consultando fuentes de datos tanto de la empresa como externas.

Otra posible mejora es la implementación de interfaces gráficas más intuitivas para el usuario, en las que se puedan definir los parámetros de forma gráfica en lugar de la introducción de los parámetros numéricamente. En la versión actual, el usuario introduce los valores en componentes de texto, sin visualizar de ninguna forma los resultados de introducir esos valores en concreto. Sin embargo, la aplicación ganaría en usabilidad si los cambios en los parámetros de frontera, numéricos, etc. se pudiesen visualizar de forma gráfica, e incluso ser editados a través de dicha visualización en lugar de la introducción numérica de sus valores.

- Sistema de paralelización: el sistema de cálculo distribuido es ineficiente en cuanto a la comunicación de datos; se pueden realizar reimplementaciones del sistema de forma que se realicen las comunicaciones de forma más óptima (por ejemplo, utilizando PVM, MPI, Spread...).

Otra mejora a realizar es la posibilidad de que las clases de resolución de problemas puedan lanzar sus propias excepciones definidas por el usuario, para poder manejar situaciones específicas de cada algoritmo.

Además, los nodos pueden mejorarse para que no se tenga que pararlos y volver a arrancarlos cuando se quiera instalar el resolutor de un nuevo tipo de problema; se puede añadir un mecanismo para que el nodo actualice la lista de problemas instalados en cualquier instante.

Para la explotación del sistema de paralelización sería necesario añadir nuevas facilidades de forma que se pudiesen ejecutar los nodos cliente de forma más sencilla, como programas (o *scripts*) nativos.

Bibliografía

- [1] A. Bermúdez, M. R. Nogueiras, C. Vázquez: *Numerical solution of variational inequalities for pricing Asian options by higher order Lagrange-Galerkin methods*, aparecerá en Applied Numerical Mathematics
- [2] A. K. Dixit and R. S. Pindyck, *Investment Under Uncertainty*, Princeton University Press, Princeton, NJ, 1994.
- [3] J. Farto and C. Vázquez, *Numerical techniques for pricing callable bonds with notice*, *Applied Mathematics and Computation*, 161 pp 989-1013, 2005
- [4] Y. d'Halluin, P.A. Forsyth and K.R. Vetzal, 2002, Managing Capacity For Telecommunications Networks Under Uncertainty. *IEEE/ACM Transactions on Networking*, 10:579-588
- [5] Y. d'Halluin, P.A. Forsyth and K.R. Vetzal, May 10th 2004, Wireless Network Capacity Management: A Real Options Approach
- [6] Y. d'Halluin, P.A. Forsyth, K.R. Vetzal and G. Labahn, *A Numerical PDE Approach For Pricing Callable Bonds*. February 2nd, 2001
- [7] Chris Kenyon and Giorgos Cheliotis, *Stochastic models for telecom commodity prices*, *Computer Networks*, 36, pp 533-555. 2001
- [8] Chris Kenyon and Giorgos Cheliotis, *European-Style Forward Derivatives for Telecom Commodities*, Research Report, April 23th, 2001
- [9] Chris Kenyon and Giorgos Cheliotis, *Forward Price Dynamics and Option Designs for Network Commodities*. Research Report, December 19th, 2001
- [10] Chris Kenyon and Giorgos Cheliotis, *Real Options and Bandwidth Markets: Dark Fiber Valuation*. July 23th, 2001
- [11] MathWorks Inc., *Matlab programming language reference*. 2002
- [12] Sun Microsystems, *The Java Tutorial*. 2001

- [13] C. Vázquez: *An upwind numerical approach for an American and European options pricing problem*, Applied Mathematics and Computation, vol 97, 273-286, 1998
- [14] P. Wilmott, *Derivatives: The Theory and Practice of Financial Engineering*, Wiley, West Sussex, England, 1998

Apéndice A

Planificación del proyecto

En el presente apéndice se muestran varias posibles planificaciones para el Proyecto Fin de Carrera.

A.1. Definición de las actividades

Las actividades que forman el proyecto son las siguientes:

1. Reunión inicial del proyecto: en esta reunión estarán se establecerá la planificación general del proyecto.
2. Lectura de los artículos de los que se extrae el modelo, y de los relacionados con el contexto que se va a tratar. Esta lectura permitirá obtener los conocimientos necesarios para realizar el desarrollo teórico de los modelos utilizados.
3. Definición del modelo de redes de acceso a Internet: en ella se define el modelo de toma de decisiones para redes de fibra óptica.
4. Definición del modelo de redes de telefonía móvil: en esta actividad se especifica el modelo matemático para la toma de decisiones de inversiones en redes de telefonía móvil.
5. Lectura de los artículos relacionados con la resolución de ecuaciones diferenciales, para adquirir los conocimientos necesarios para la definición del método de elementos finitos.
6. Definición del método de elementos finitos con características: en la que se desarrolla todo el método numérico utilizado en la resolución de las ecuaciones implicadas.
7. Construcción del prototipo de resolución del método numérico.
8. Construcción del prototipo de toma de decisiones para redes de fibra óptica.
9. Construcción del prototipo de toma de decisiones para redes de telefonía móvil.

10. Construcción de la aplicación, entendiendo ésta como la infraestructura general en la que se incluirán los módulos que se desarrollen posteriormente.
11. Construcción del módulo para redes de acceso a Internet, e inclusión del mismo en la aplicación software.
12. Construcción del módulo para redes de telefonía móvil, e inclusión del mismo en la aplicación software.
13. Construcción del sistema de cálculo distribuido.
14. Inclusión del sistema de cálculo distribuido en la toma de decisiones en redes de fibra óptica, modificando el módulo ya construido.
15. Inclusión del sistema de cálculo distribuido en la toma de decisiones en redes de telefonía móvil, modificando el módulo ya construido.

A.2. Restricciones entre actividades

Las actividades no pueden ser ejecutadas en paralelo, debido a las dependencias que existen entre ellas. Estas dependencias se muestran en la Figura A.1. Todas las restricciones mostradas son del tipo FF (*Finish to start*) por lo que ninguna de las actividades puede comenzar su ejecución hasta que no hayan acabado todas sus actividades predecesoras.

A.3. Planificaciones

Así, se puede plantear la planificación inicial mostrada en la figura A.2. En el proyecto actual el personal disponible consiste en una única persona en los cargos de analista, diseño y programador, y otra en el rol de director de proyecto. Esto hace que la planificación real del proyecto sea secuencial de forma obligatoria, ya que sólo hay una persona realizando las tareas del proyecto con dedicación completa, por lo que el camino crítico estará formado por todas las actividades enumeradas. Por ello, no importa el orden en el que se realicen (manteniendo por supuesto el orden impuesto por las restricciones entre tareas). Un retraso en cualquiera de las actividades se traducirá en un retraso en la fecha de fin de proyecto. La duración de la planificación es de aproximadamente 7 meses, que conforma la duración real estimada del proyecto.

Se puede plantear una planificación con más recursos, en la que existen dos equipos de desarrollo de software; esto permite que algunas tareas se puedan ejecutar de forma paralela. Cada equipo se considera compuesto por una persona, capaz de realizar tareas de análisis, diseño e implementación. Esta planificación está mostrada en la Figura A.3. Como es lógico, esta planificación no es real, ya que para

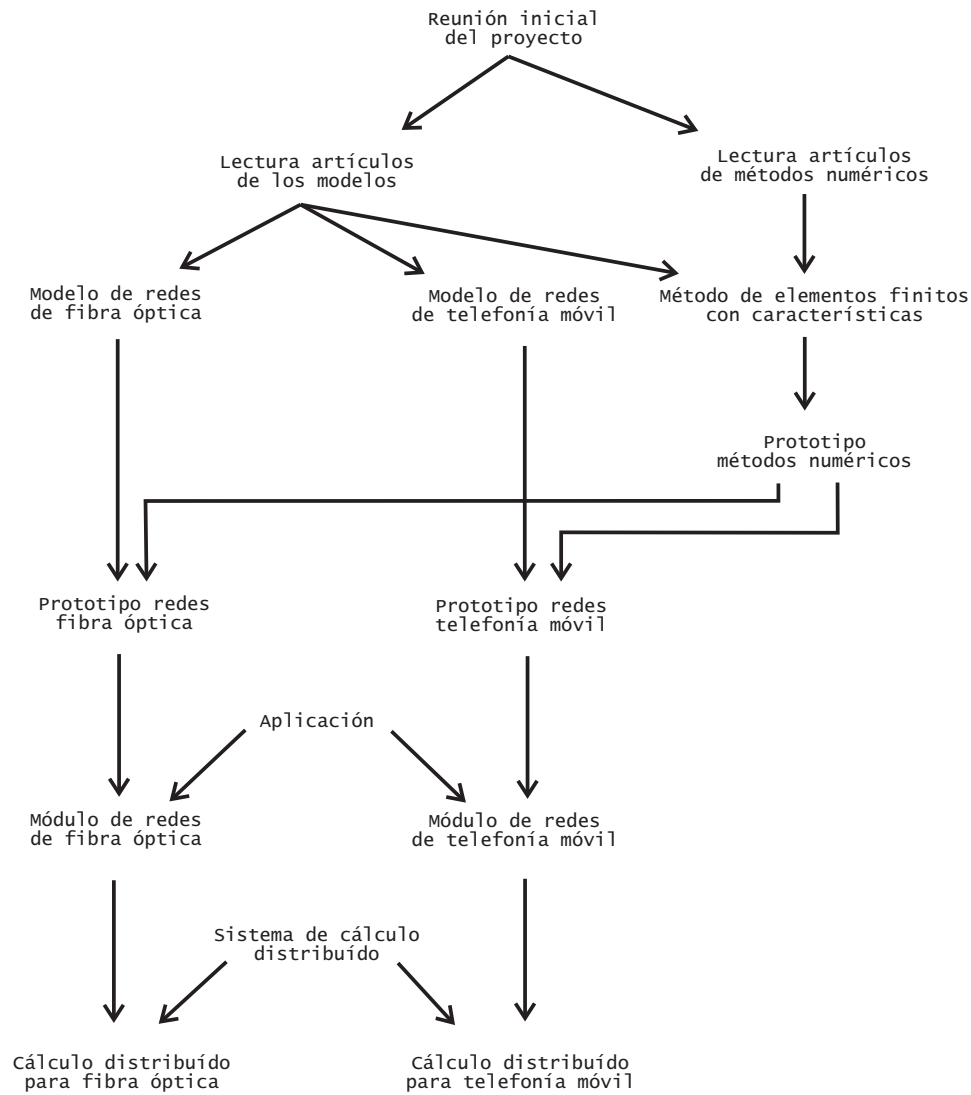


Figura A.1: Las restricciones entre las actividades del proyecto

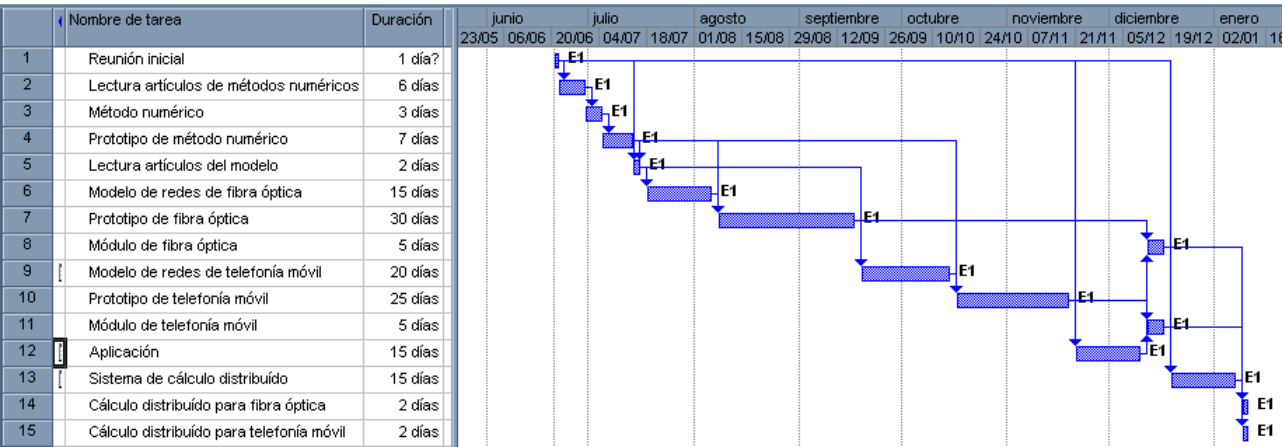


Figura A.2: La planificación del proyecto

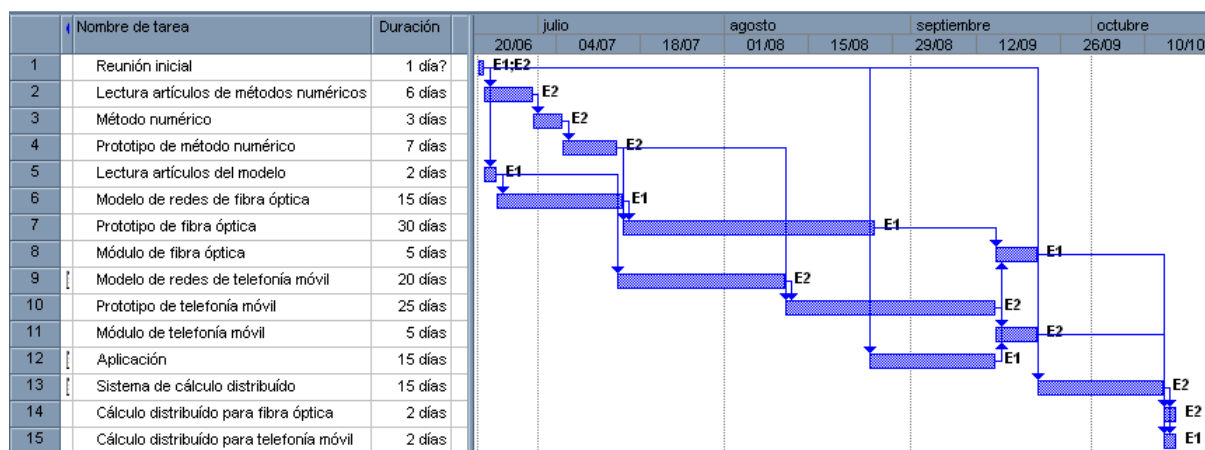


Figura A.3: Una posible planificación del proyecto

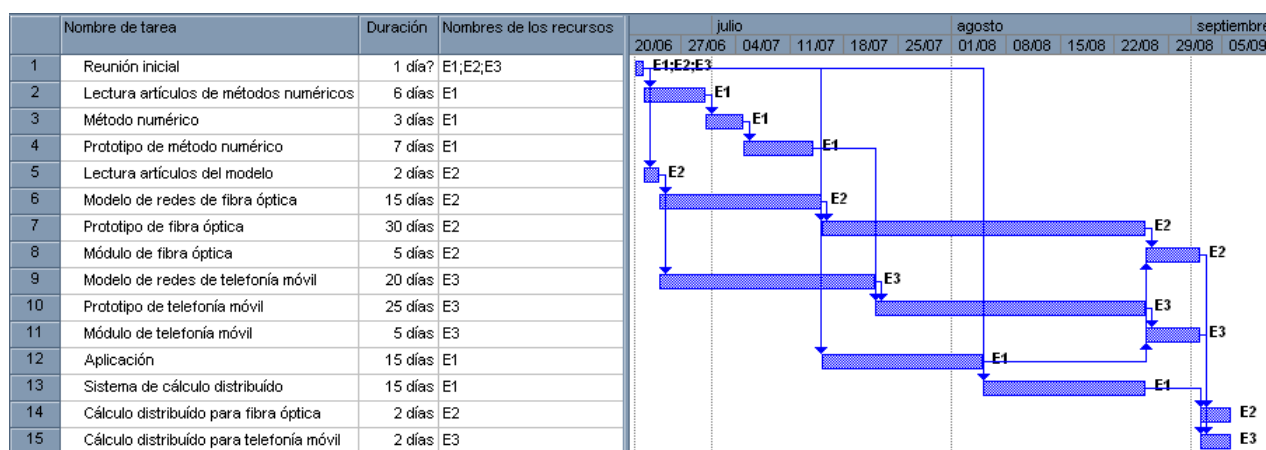


Figura A.4: La mejor planificación temporal del proyecto

la consecución del presente Proyecto Fin de Carrera no se dispone de varios equipos de desarrollo, sino solamente de uno. Con esta nueva planificación se reduce a aproximadamente 4 meses la duración del proyecto.

Sin embargo, si se considera un equipo más de desarrollo, se puede conseguir la planificación mostrada en la Figura A.4. Esta planificación es la mejor posible en cuanto a utilización de tiempo: efectivamente, aprovecha las dependencias entre tareas de la forma que obtiene el mayor paralelismo posible en la ejecución de las actividades. Sin embargo, es la planificación que más recursos necesitará, dado que el realizar varias tareas al mismo tiempo hace que las tengan que realizar personal diferente, mientras que ejecutándolas de forma secuencial las puede realizar una única persona. Obviamente, esta planificación también es ficticia, y permite realizar el proyecto en 2 meses y medio, aproximadamente.