Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA

TESIS DOCTORAL

Aplicación de Técnicas de Aprendizaje Automático a la Gestión y Optimización de Cachés de Teselas para la Aceleración de Servicios de Mapas en las Infraestructuras de Datos Espaciales

Presentada por RICARDO GARCÍA MARTÍN para optar al grado de doctor por la Universidad de Valladolid

Dirigida por:

Juan Pablo de Castro Fernández

María Jesús Verdú Pérez

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA

TESIS DOCTORAL

Aplicación de Técnicas de Aprendizaje Automático a la Gestión y Optimización de Cachés de Teselas para la Aceleración de Servicios de Mapas en las Infraestructuras de Datos Espaciales

Presentada por RICARDO GARCÍA MARTÍN para optar al grado de doctor por la Universidad de Valladolid

Dirigida por:

Juan Pablo de Castro Fernández

María Jesús Verdú Pérez

A mis abuelos, Félix y Noé

Agradecimientos

Durante estos años han sido muchas las personas e instituciones que han participado en este trabajo y a quienes quiero expresar mi gratitud por el apoyo y la confianza depositados.

Comenzando por éstas últimas, he de mencionar el respaldo del Ministerio de Ciencia e Innovación, que a través del proyecto CENIT España Virtual[1] (ref. CENIT 2008-1030) del programa Ingenio 2010[2], cofinanciado por parte del Centro Nacional de Información Geográfica (CNIG) y el Centro para el Desarrollo Tecnológico Industrial (CDTI), ha dado lugar a un marco inmejorable para el desarrollo de esta investigación.

Debo también un especial reconocimiento a la Universidad de Valladolid por concederme una beca para personal investigador en formación (PIF-UVa).

Gracias al Instituto Geográfico Nacional (IGN) por ceder los registros de peticiones a sus servidores de mapas y por su apoyo a esta investigación a través de convenios de colaboración. Quiero hacer una mención especial a Antonio F. Rogríguez, uno de los principales impulsores de las Infraestructuras de Datos Espaciales en nuestro país.

Por otra parte, quiero expresar mi más sincero agradecimiento a los Doctores Juan Pablo de Castro y María Jesús Verdú, director y codirector de esta tesis, por toda la atención, estímulo y experta dirección, tanto profesional como humana, prestados durante el tiempo de realización de este trabajo.

Asimismo agradecer a mis otros compañeros del grupo de investigación, Luisa M. Regueras y Elena Verdú, por participar en la revisión técnica de los trabajos, y por la creación de un ambiente de trabajo tan agradable.

Por último, aunque no menos importante, agradecer a Jordana, a mis hermanos - Víctor y Dani - y a mis padres - Ángel y Gabi - su apoyo siempre incondicional.

[1] http://www.españavirtual.org
[2] http://www.ingenio2010.es

Resumen

La gran proliferación en el uso de servicios de mapas a través de la Web ha motivado la necesidad de disponer de servicios cada vez más escalables. Como respuesta a esta necesidad, los servicios de mapas basados en teselado se han perfilado como una alternativa escalable frente a los servicios de mapas tradicionales, permitiendo la actuación de mecanismos de caché o incluso la prestación del servicio mediante una colección de imágenes pregeneradas. Sin embargo, los requisitos de almacenamiento y tiempo de puesta en marcha de estos servicios resultan a menudo prohibitivos cuando la cartografía a servir cubre una zona geográfica extensa para un elevado número de escalas.

Por ello, habitualmente estos servicios se ofrecen recurriendo a cachés parciales que contienen tan solo un subconjunto de la cartografía. Para garantizar una Calidad de Servicio (QoS - *Quality of Service*) aceptable es necesaria la actuación de adecuadas políticas de mantenimiento y gestión de estas cachés de mapas: 1) Estrategias de población inicial ó *seeding* de la caché. 2) Algoritmos de carga dinámica ante las peticiones de los usuarios. 3) Políticas de reemplazo de caché.

Sin embargo, existe un reducido número de estas estrategias que sean específicas para los servicios de mapas. La mayor parte de estrategias aplicadas a estos servicios son extraídas de otros ámbitos, como los proxies Web tradicionales, las cuáles no tienen en cuenta la componente espacial de los objetos de mapa que gestionan.

En la presente tesis se aborda este punto de mejora, diseñando nuevos algoritmos específicos para este dominio de aplicación que permitan optimizar el rendimiento de los servicios de mapas. Dado el elevado número de objetos gestionados por estas cachés y la heterogeneidad de los mismos en cuanto a capas, escalas de representación, etc., se ha hecho un esfuerzo para que las estrategias diseñadas sean automáticas o semi-automáticas, requiriendo la menor intervención humana posible.

Así, se han propuesto dos novedosas estrategias para la población inicial de una caché de mapas. Una de ellas utiliza un modelo descriptivo mediante los registros de peticiones pasadas del servicio. La otra se basa en un modelo predictivo para la identificación de fenómenos geográficos directores de las peticiones de los usuarios, parametrizado o bien mediante un análisis regresivo OLS (*Ordinary Least Squares*) o mediante un sistema inteligente con redes neuronales.

Asimismo, se han llevado a cabo importantes contribuciones en relación con las estrategias de reemplazo de estas cachés. Por una parte, se ha propuesto un sistema inteligente basado en redes neuronales, que estima la popularidad de acceso futuro en base a ciertas propiedades de los objetos que gestiona: actualidad de referencia, frecuencia de referencia, y el tamaño de la tesela referenciada. Por otra parte, se ha propuesto una estrategia, bautizada como *Spatial-LFU*, la cual es una variante de la estrategia *Perfect-LFU*, simplificada aprovechando la correlación espacial existente entre las peticiones.

Además, se ha propuesto una nueva estrategia de carga dinámica mediante *Metatiling* adaptativo, el cuál elige la zona a precargar en función del estado de la caché.

Las estrategias anteriores han sido validadas utilizando registros de peticiones de diversos servicios de mapas públicos de ámbito nacional. Para ello, se han implementado diversos simuladores y prototipos, liberados ahora a la comunidad.

Abstract

The increasing popularity of web map services has motivated the development of more scalable services in the Spatial Data Infrastructures (SDI). Tiled map services have emerged as a scalable alternative to traditional map services. Instead of rendering map images on the fly, a collection of pre-generated image tiles can be served very fast from a server-side cache. However, these caches can grow unmanageably in size when the cartography covers mid to large areas for multiple rendering scales, so storage requirements and start-up time are often prohibitive for many organizations.

This forces modest organizations to use partial caches containing just a subset of the total tiles. In order to guarantee and acceptable *Quality of Service* (QoS), the intervention of adequate cache management policies is required: 1) Initial population or cache seeding. 2) Dynamic tile prefetching upon user's requests. 3) Cache replacement policies.

However, there is a reduced number of cache management strategies specific to map tile caches. Most caching strategies applied to web map services are extracted from other fields, such as main memory management, which do not benefit of the spatial component of its managed objects. This thesis addresses this point of improvement by designing new algorithms, for this application domain, that optimize the performance of web map services.

Given the high volume of the objects managed by these caches and the heterogeneity as regards layers, representation scales, etc., a big effort has been made to automatize these processes as far as possible, requiring minimum human intervention.

This way, two novel seeding strategies for the initial population of map caches have been proposed. The first one uses a descriptive model based on the mining of web server logs. The second one is based on the use of a predictive model for the identification of interesting geographic features that guide most of user's attention, parameterized either by OLS regression analysis or by an intelligent system that uses neural networks.

In the same way, significant contributions regarding to cache replacement policies have been presented. In the one hand, a neural network based intelligent system has been proposed. This system estimates the future probability of tiles based on certain properties of web objects: recency, frequency and size. On the other hand, a variation of the popular *Perfect-LFU* replacement strategy, named *Spatial-LFU* has been proposed. This reduces the overhead of storing the statistics of all tiles by taken into account the spatial auto-correlation between requests.

Moreover, a new prefetching strategy, based on adaptive *metatiling* has been proposed. This proposal selects the prefetching area according to current state of the cache.

The previous strategies have been validated using trace files extracted from multiple nation-wide public web map services in Spain. Several simulators and prototypes have been implemented, published under Open Source license, in order to experiment with the proposed strategies.

Índice

Agradecimientos vii

Resumen ix

Abstract xi

1. Introducción 1
- 1.1. Objetivos de la tesis y contribuciones . . . 4
- 1.2. Metodología . . . 5
- 1.3. Estructura de la memoria . . . 7

2. Servicios Web de Mapas 9
- 2.1. Web Map Service (WMS) . . . 9
- 2.2. Servicios de Mapas Teselados . . . 13
  - 2.2.1. Propuesta WMS Tile Caching (WMS-C) de OSGeo . . . 15
  - 2.2.2. Web Map Tile Service (WMTS) . . . 16
  - 2.2.3. Microsoft Bing Maps y Google Maps . . . 19
- 2.3. Clientes de Mapas . . . 20

3. Gestión y optimización de las cachés de servicios OGC teselados 23

4. Estado de arte 31
- 4.1. Organización y traspaso de los datos . . . 31
- 4.2. Almacenamiento de las teselas . . . 33
  - 4.2.1. Almacenamiento en sistema de ficheros . . . 33
  - 4.2.2. Almacenamiento en bases de datos relacionales . . . 34
  - 4.2.3. Almacenamiento en la nube . . . 34
  - 4.2.4. Almacenamiento distribuido P2P . . . 35
- 4.3. Compresión de las imágenes de mapa . . . 35
- 4.4. Políticas de gestión de la caché . . . 36
  - 4.4.1. Políticas de precarga inicial (*long-term*) . . . 36
  - 4.4.2. Políticas de carga dinámica (*short term*) . . . 37
  - 4.4.3. Políticas de reemplazo . . . 41

5. Caracterización de tráfico de servicios de mapas teselados 43
- 5.1. Registros de Acceso (Logs) . . . 44

5.2. Servicios de Mapas Analizados . . . 45
5.3. Datos crudos . . . 46
5.4. Reducción de los datos . . . 47
5.5. Concentración de las peticiones . . . 47
5.6. Distribución de peticiones por escala . . . 49
5.7. Distribución del Tamaño de las Teselas . . . 50
5.8. Análisis temporal . . . 55
5.9. Tiempos entre llamadas sucesivas de un mismo objeto . . . 58
5.10. Autocorrelación espacial . . . 59
5.11. Correlación espacial cruzada . . . 63
5.12. Origen geográfico de las peticiones . . . 65
5.13. Aplicación al resto de la tesis . . . 67

6. Políticas de reemplazo de caché 69
6.1. Estrategia de reemplazo de caché mediante redes neuronales . . . 70
6.1.1. Entradas de la red neuronal . . . 71
6.1.2. Salida de la red neuronal . . . 73
6.1.3. Entrenamiento de la red neuronal . . . 74
6.1.4. Resultados . . . 74
6.2. Estrategia de reemplazo de caché Spatial-LFU . . . 76
6.2.1. Comparativa de rendimiento entre las estrategias de reemplazo más populares . . . 77
6.2.2. Reducción de la sobrecarga de los algoritmos LFU . . . 77
6.2.3. Resultados . . . 79

7. Políticas de precarga de caché 85
7.1. Arquitectura del Sistema . . . 87
7.1.1. Proxy de teselas . . . 87
7.1.2. Gestor de caché . . . 88
7.1.3. Catálogo de fenómenos geográficos . . . 88
7.1.4. Selector de fenómenos geográficos . . . 89
7.1.5. Extractor del modelo . . . 89
7.1.6. Planificador de la tarea de precarga . . . 89
7.2. Modelo Descriptivo . . . 89
7.3. Modelo Predictivo . . . 101
7.3.1. Metodología . . . 102
7.3.2. Modelo predictivo basado en regresiones lineales . . . 103
7.3.3. Modelo predictivo basado en redes neuronales . . . 113
7.4. Análisis comparativo frente a los sistemas de gestión de carga dinámica . . . 119

8. Políticas de carga dinámica 121
8.1. Rendimiento del metatiling en la tarea de seeding . . . 125
8.2. Rendimiento del metatiling durante la carga dinámica . . . 126

9. Conclusiones y trabajos futuros 129
9.1. Conclusiones . . . 129

9.2. Futuras Líneas de Investigación 131

A. Prototipos de caché de teselas **135**
A.1. Comparativa de Implementaciones de Caché 135
A.2. Prototipo de caché de teselas WMSCWrapper 137
A.2.1. Arquitectura 138
A.2.2. Configuración del servicio 147
A.2.3. Estrategia de monitorización 148
A.2.4. Mecanismos de precarga de teselas y de limpieza de la caché 148
A.2.5. Benchmarking del servicio de caché 157
A.3. Extensión del prototipo de caché de teselas GeoWebCache 163
A.3.1. Implementación de la solución 163
A.3.2. Módulo para el almacenamiento de estadísticas 164
A.3.3. Módulo de catálogo de fenómenos geográficos 165
A.3.4. Módulo de precarga de teselas 166
A.3.5. Posibles puntos de mejora 169
A.4. Simulador para la evaluación de políticas de reemplazo de una caché de mapas 169

B. Trabajos Relacionados con la Investigación **173**
B.1. Publicaciones derivadas de la investigación 173
B.1.1. Revista 173
B.1.2. Capítulos de libro 174
B.1.3. Congresos internacionales 174
B.1.4. Congresos nacionales 175
B.2. Publicaciones no directamente relacionadas 178
B.2.1. Revista 178
B.2.2. Congresos internacionales 178
B.2.3. Congresos nacionales 179
B.3. Participación en Proyectos de Investigación 179
B.4. Becas de investigación 180
B.5. Dirección de Proyectos Fin de Carrera 181

Bibliografía **183**

Lista de acrónimos **197**

Índice de figuras

1.1. Proxy WMS-C en la IDE. 2
1.2. Contexto, objetivos y contribuciones de la tesis doctoral. 6

2.1. Modelo OpenGIS del *workflow* para la representación de una imagen de mapa. 10
2.2. Ortofoto de la ETSIT de Valladolid obtenida a partir del servicio WMS del PNOA. 13
2.3. Imagen de mapa vectorial con cartografía catastral de la ETSIT y alrededores obtenida a través del servicio WMS de la Dirección General del Catastro. . 14
2.4. Ortofoto de la ETSIT de Valladolid obtenida a partir del servicio WMS del PNOA. 14
2.5. Estructura piramidal de teselas para una capa expuesta a través de un servicio WMTS. 17
2.6. Espacio de teselado definido por el estándar WMTS. 18
2.7. Proyección Mercator utilizada por Bing Maps. 19
2.8. Esquema de teselado en Bing Maps para la escala 3. 20
2.9. *Quadkeys* en Bing Maps. 21
2.10. Teselas cargadas por el cliente OpenLayers para la región visualizada con distintos valores de *búffer* de pre-carga. 22

3.1. Taxonomía de los sistemas y puntos de actuación de las caché. 24
3.2. Relación general de un filtro tipo *Web Cache* con los servicios OGC generadores de objetos a partir de parámetros espaciales discretizables. 25
3.3. Espacio uniforme de teselado en un nivel concreto de la pirámide de escalas. 27
3.4. Método para extrapolar características probabilísticas usando el principio de localidad. 30

4.1. Descomposición de un mapa en capas temáticas. 32
4.2. Curvas de relleno del espacio. Fuente: (Rigaux et al., 2002). 34
4.3. *Workflow* para determinar las áreas prioritarias para el cacheo. 37
4.4. Arquitectura de un servicio Web GIS. 39

5.1. Escenario para la extracción, normalización y almacenamiento de los datos procedentes de los históricos de peticiones a los servidores de mapas. 45
5.2. Función de distribución acumulada (CDF) de accesos a las teselas. 48
5.3. Función de distribución acumulativa complementaria (CCDF) de referencias a las teselas y su ajuste por ley de potencias. 50

5.4. Distribución normalizada de las peticiones en función del nivel de resolución para los diversos servicios bajo estudio. 51
5.5. Distribución del tamaño de las teselas para los diferentes servicios analizados. 52
5.6. Correlación entre la frecuencia de acceso a las teselas y su tamaño para cada escala de representación. 53
5.7. Tamaño medio de los objetos frente a su frecuencia de referencia. 54
5.8. Distribución de las peticiones de los usuarios en función del día de la semana y la hora. 56
5.9. Distribución de las peticiones de los usuarios en función de la hora del día. . 57
5.10. Distribución de las peticiones de los usuarios en función del día de la semana. 57
5.11. Estacionariedad de las peticiones en PNOA. Madrid capital. 58
5.12. Distribución de tiempos entre peticiones sucesivas de un mismo objeto. . . . 59
5.13. Distribución de probabilidad d(k) de recibir otra petición del mismo objeto k peticiones después. 60
5.14. Misma cuadrícula para dos escalas de resolución. a) raster de 4x4 (izda), b) raster de 8x8 (dcha). 62
5.15. Medidas de autocorrelación espacial de Moran&Geary de las peticiones de teselas para cada escala en los servicios WMS-C de Cartociudad e IDEE-Base. 63
5.16. Correlograma de Moran para el servicio IDEE-Base, escala 10. 64
5.17. Correlación bivaluada de Moran. 65
5.18. Posición geográfica de los clientes del servicio IDEE-Base obtenidas a partir de los logs de acceso. 66
5.19. Relación entre la posición geográfica de los clientes del servicio IDEE-Base y el destino geográfico de sus peticiones. 66

6.1. Neurona artificial. 70
6.2. Red neuronal propuesta, de tipo *feed-forward* con dos capas ocultas. 71
6.3. CCR obtenidos para Cartociudad. 75
6.4. CCR obtenidos para IDEE-Base. 75
6.5. CCR obtenidos para PNOA. 76
6.6. Comparativa del porcentaje de aciertos de caché usando los algoritmos InCache-LFU, Perfect-LFU, LRU y el algoritmo óptimo de Belady OPT. 78
6.7. Comparativa del porcentaje de aciertos de caché usando los algoritmos InCache-LFU Simplificados frente a InCache-LFU y LRU. 80
6.8. Consumo de recursos para estadísticas con el algoritmo InCache-LFU Simplificado, para distintos factores de simplificación. Tamaño de la caché: 209715200 bytes. 81
6.9. Consumo de recursos para estadísticas con el algoritmo InCache-LFU Simplificado, para distintos factores de simplificación. Tamaño de la caché: 6553600 bytes. 82
6.10. Comparativa del porcentaje de aciertos de caché usando los algoritmos Perfect-Spatial-LFU frente a Perfect-LFU e InCache-LFU. 83
6.11. Comparativa del consumo de recursos usando los algoritmos Perfect-Spatial-LFU frente a Perfect-LFU. 83

7.1. Arquitectura propuesta para la precarga de teselas a partir de un catálogo general de fenómenos geográficos y la historia del servicio. 87
7.2. Visualización de patrones de acceso en Microsoft Hotmap. 90
7.3. Región considerada (área sombreada) para el estudio. 91
7.4. Transformación "conceptual" de una pirámide de escalas a un prisma mediante el uso del modelo simplificado. 92
7.5. Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de Cartociudad propagadas al nivel de resolución 12. 93
7.6. Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de PNOA propagadas al nivel de resolución 12. 94
7.7. Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de IDEE-BASE propagadas al nivel de resolución 12. 95
7.8. Espacio de almacenamiento de una tesela para Cartociudad (izquierda), PNOA (centro) e IDEE-Base (derecha), cubriendo una zona de mar (arriba) y una urbana (abajo). 97
7.7. % Aciertos de caché Vs % de objetos cacheados en IDEE. 100
7.8. Mapa de calor de las peticiones recibidas en el área de estudio por el servicio PNOA en el nivel de resolución 18, entre el 19 de Marzo de 2010 y el 17 de Junio del mismo año. Lás zonas más oscuras corresponden a las teselas que han recibido un mayor número de solicitudes por parte de los usuarios. Para mayor claridad, la representación se realiza en escala logarítmica. 103
7.9. Catálogo de fenómenos geográficos utilizado para simulación del modelo regresivo. 106
7.10. Coeficientes de la regresión OLS en PNOA, para diferentes períodos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. Los coeficientes están normalizados de forma que la suma de los coeficientes positivos es igual a 1. 108
7.11. Coeficientes de la regresión OLS en IDEE-Base, para diferentes períodos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. Los coeficientes están normalizados de forma que la suma de los coeficientes positivos es igual a 1. 109
7.12. Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Cache-Hit Ratio (CHR), obtenidas en PNOA para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. 110
7.13. Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Cache-Hit Ratio (CHR), obtenidas en IDEE-Base para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. 111
7.14. Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Byte-Hit Ratio (BHR), obtenidas en PNOA para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. 112
7.15. Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Byte-Hit Ratio (BHR), obtenidas en IDEE-Base para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. 112

7.16. Arquitectura de ANN de tipo 19/6/1 utilizada en este estudio. La función de activación de las neuronas de la capa oculta es de tipo tangente-sigmoidea, mientras que en la capa de salida se utiliza una función lineal. 115
7.17. Comparativa de rendimiento conseguido en PNOA en términos de *Cache-Hit Ratio* para un tiempo de entrenamiento de 1 día. 117
7.18. Comparativa de rendimiento conseguido en PNOA en términos de *Byte-Hit Ratio* para un tiempo de entrenamiento de 1 día. 118
7.19. Evolución temporal de la tasa de acierto y del consumo de recursos del servicio PNOA mediante el procedimiento de carga dinámica con las peticiones de los usuarios. 120

8.1. Flujo de una petición de mapa con metatiling. 121
8.2. Problema del etiquetado redundante en un servicio de mapas. (a) Al pedir teselas individuales aparecen etiquetas duplicadas entre teselas adyacentes. (b) Con metatiling no se producen duplicados de etiquetas. 122
8.3. Analogía de la transferencia de información por bloques entre una jerarquía memoria de un ordenador y un servicio de mapas. 123
8.4. Estrategia de mínima correlación con la caché para la generación de *metatiles*. 124
8.5. *Metatiles* pedidos al servidor de mapas remoto durante una tarea de *seeding*, utilizando *metatiles* centradas en la tesela a cachear, con $B = 2$. 126
8.6. *Metatiles* pedidos al servidor de mapas remoto durante una tarea de *seeding*, utilizando la estrategia de *metatiling* de mínima correlación con la caché, con $B = 2$. 127
8.7. Comparativa de objetos cacheados al finalizar la tarea, para distintos tamaños de *buffer* y configuraciones de *metatile*, tras 1.000.000 de peticiones realizadas a la caché. 128
8.8. Comparativa de aciertos de caché, para distintos tamaños de *buffer* y configuraciones de *metatile*, tras 1.000.000 de peticiones realizadas a la caché. . . 128

A.1. Configuración multi-módulo del proyecto Maven de WMSCWrapper. 138
A.2. Diagrama de componentes (simplificado) de la caché WMSCWrapper. . . . 139
A.3. Captura de Google Earth con cartografía obtenida de la caché WMSC-Wrapper en modo *debug*. 140
A.4. Captura de Google Maps con cartografía obtenida de la caché WMSCWrapper en modo *debug*. 141
A.5. Captura de Bing Maps con cartografía obtenida de la caché WMSCWrapper en modo *debug*. 142
A.6. Estructura de directorios para el almacenamiento persistente de las teselas en WMSCWrapper. 143
A.7. Recopilación de estadísticas en el índice espacial 145
A.8. Capa de teselas virtuales (*StatsLayerSet*) representando una de las estadísticas recopiladas junto con información de depuración. 146
A.9. Selección de zonas geográficas en ArcGis para la ejecución de una tarea de *seeding*. 149
A.10. Interfaz gráfica de usuario para la selección de geometrías en el mapa. . . . 150
A.11. Generación y consumo de *heatmaps* para explotación en la caché. 151

A.12.Estrategia de generación de *heatmap* a través de las *features* de un WFS según la aproximación *up-down*. 152
A.13.Estrategia de generación de *heatmap* a través de las *features* de un WFS según la aproximación *bottom-up*. 153
A.14.Capa vectorial con los segmentos de los ríos de la IDE de la Confederación Hidrográfica del Duero. 154
A.15.Capa *CachedLayerSet* con el contenido cacheado tras ejecutar una tarea de precarga con la aproximación *Bottom-Up*. Las teselas sombreadas no están en caché. 155
A.16.Mapa de estimación de probabilidad *a posteriori* calculado a partir de los registros de Cartociudad. Accedido por un cliente de mapas *WorldWind* a través de un servicio WMS estándar. 156
A.17.Identificación de cambios de gradiente mediante procesado de imagen para la generación de ROI (*Regions of Interest*). 157
A.18.Detección de Regiones de Interés (RoI) mediante *clustering*. Agrupación en 4 *clusters* 158
A.19.Arquitectura del entorno de pruebas 159
A.20.Definición de los parámetros de la petición WMS-C en JMeter 160
A.21.Resultados de las medidas de rendimiento en términos de *throughput* de los sistemas de caché TileCache, GeoWebCache y WMSCWrapper para distinto número de hilos. Caché inicialmente llena. 161
A.22.Resultados de las medidas de rendimiento en términos de *throughput* de los sistemas de caché TileCache, GeoWebCache y WMSCWrapper para distinto número de hilos. Caché inicialmente vacía. 161
A.23.Tiempos de respuesta para distintos tamaños de *metatile*, con caché inicialmente vacía. 162
A.24.Tasa de transmisión para distintos tamaños de *metatile*, con caché inicialmente vacía. 162
A.25.Organización del proyecto Maven de GeoWebCache. 164
A.26.Organización del proyecto Maven para la implementación propuesta. 165
A.27.Componentes principales del módulo gwc-stats (fichero geowebcache-stats-context.xml). 166
A.28.Diagrama de secuencia para el almacenamiento de estadísticas. 166
A.29.Clases de GeoTools para el acceso a fuentes de fenómenos. 167
A.30.Catálogo de fenómenos y componente de precarga. 168
A.31.Captura del simulador de estrategias de reemplazo. 170

Índice de Tablas

2.1. Parámetros de una petición *GetMap* 11
2.2. Petición WMS GetMap al servicio del PNOA que devuelve la imagen de mapa de la Figura 2.2. 12
2.3. Petición WMS GetMap al servicio del Catastro que devuelve la imagen de mapa de la Figura 2.3. 12

5.1. Parámetros de interés de los servicios WMS-C analizados. 46
5.2. Estadísticas globales de los registros de peticiones a los servicios WMS-C . . 47
5.3. Tiempos medios entre referencias sucesivas de un mismo objeto. 58

6.1. Parámetros de la red neuronal propuesta. 72
6.2. CCR (%) obtenidos durante las fases de entrenamiento, y validación y test para los distintos servicios 74

7.1. Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de Cartociudad, con μ =media. 96
7.2. Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de IDEE, con μ =media. 96
7.3. Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de PNOA, con μ =media. 96
7.4. Coeficientes de la combinación lineal que minimiza el error cuadrático medio para la predicción de las peticiones de los usuarios en un determinado nivel a partir de las estadísticas recogidas en otros niveles en IDEE-Base. 101
7.5. Tasa de acierto de caché y consumo de recursos utilizando una combinación lineal que minimiza el error cuadrático medio (LSM) para la predicción de las peticiones de los usuarios para cada nivel a partir de las estadísticas recogidas en los otros niveles en IDEE-Base. 101
7.6. Fenómenos geográficos del catálogo, con sus respectivos tipos de geometría y el porcentaje de teselas cubiertos por cada fenómeno al nivel de resolución 15.107
7.7. Número de peticiones registradas en los conjuntos de datos usados para simulación en PNOA e IDEE en los niveles de resolución 18 y 16, respectivamente.108
7.8. Parámetros de la red neuronal propuesta. 116

8.1. Latencias medias para la obtención de un objeto a partir del servicio WMS original (CORINE) en función del tamaño del *metatile*. 124

A.1. Resumen de características de las principales implementaciones de caché de teselas de código abierto. 136

Capítulo 1

Introducción

Las diversas especificaciones *Web Map Service* (WMS) del *Open Geospatial Consortium* (OGC) (de la Beaujardiere, 2006) ofrecen una gran flexibilidad en el servicio. Sin embargo, los parámetros espaciales de las peticiones no están restringidos, lo que hace que cada petición de mapa deba ser atendida en tiempo real mediante un procedimiento, generalmente costoso, que implica acceso a datos de origen, aplicación de estilos, composición de capas y codificación de la imagen comprimida.

Este procedimiento se ha demostrado ineficaz para satisfacer la demanda de algunas aplicaciones de difusión masiva como se expone en (Plesea, 2008) tras la experiencia del servidor OnEarth de la NASA. Por este motivo, generalmente los servicios comerciales más populares se prestan con servidores no OGC en los que el espacio geográfico está teselado de acuerdo a una rejilla predefinida y cuyo contenido está frecuentemente pregenerado (Bell et al., 2007).

La popularidad y necesidad de esta estrategia ha provocado la aparición de algunos mecanismos de acceso no estándar (Přidal, 2008; OSGeo, 2008a; Schwartz, 2009), y otras recomendaciones más abiertas como la recomendación de WMS-C (*Web Map Service - Cached*) de OSGeo (OSGeo, 2008b) o el posterior estándar WMTS del propio OGC (Masó et al., 2010).

Cuando un servicio OGC se va a utilizar en un escenario exigente con una información poco parametrizable y estacionaria, se puede utilizar el patrón *proxy web cache* para conseguir una mejora en la calidad de servicio. El *proxy* es un dispositivo que se interpone de forma preferiblemente "transparente" entre el cliente y el servicio final, como se muestra en la Figura 1.1.

Al adoptar esta aproximación, los proveedores tienen que afrontar serias decisiones relativas al diseño y mantenimiento de sus cachés de teselas. Una opción es pregenerar todas las posibles teselas en todas las escalas soportadas. Hoy en día, tan sólo las grandes corporaciones que ofrecen estos servicios disponen de las ingentes cantidades de espacio de almacenamiento y tiempo de computación necesarios para poder permitirse el tener pregenerados todos los objetos. No supone un problema para estas entidades (sí para otras más humildes y de menor presupuesto) el discernir qué contenido debe ser pregenerado y cuál no.

Para ilustrar la magnitud del problema, considérese por ejemplo que para precargar por completo en *GeoWebcache*[1] la capa de hidrografía nacional, descargable a través del

[1] http://geowebcache.org

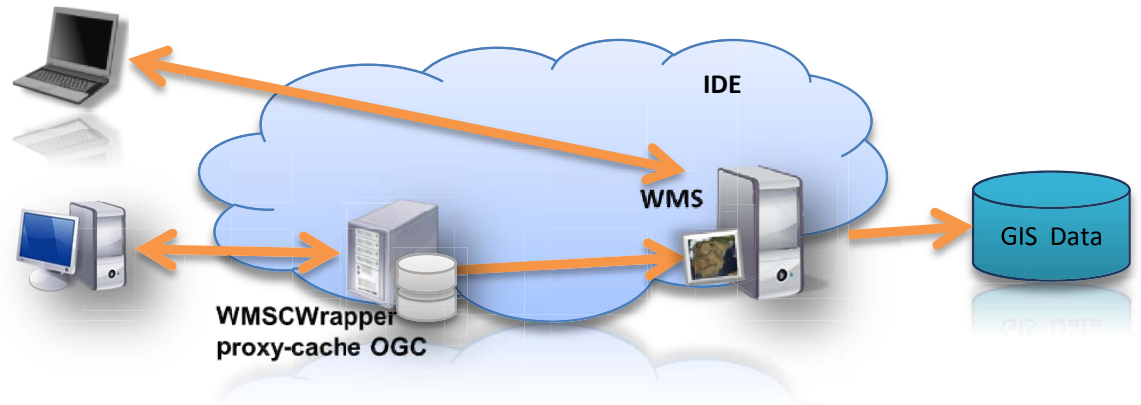


Figura 1.1: Proxy WMS-C en la IDE.

portal de las Infraestructuras de Datos Espaciales de España (IDEE)[2], hasta la escala 15 utilizando un único hilo de ejecución en un PC convencional (AMD Athlon 64, Core 2 Duo, 3 Ghz, 3 GB de RAM) el tiempo estimado es de 117 días, y se requiere un espacio en disco de 2.77 TeraBytes, aproximadamente.

En cualquier caso, incluso contando con espacio de almacenamiento suficiente, hay capas cartográficas que deben actualizarse eventualmente y todas deben, en algún momento, partir de cero o renovar su base de teselas. En general, esta estrategia de pregenerar todos los objetos no es adecuada para servir mapas que se actualizan con mucha frecuencia, como es el caso de los mapas meteorológicos o aquellos que muestran información de tráfico rodado, por citar algunos ejemplos (Loechel y Schmid, 2013).

Estos escenarios requieren una nueva aproximación basada en una estrategia que equilibre cuidadosamente los procesos de población y mantenimiento de la caché. Estos problemas de optimización de caché se resuelven con frecuencia mediante heurísticas como las siguientes: (1) monitorizar la popularidad de teselas individuales, y (2) cachear las teselas más solicitadas. Esta aproximación presenta, sin embargo, los siguientes inconvenientes. En primer lugar, asume que los usuarios tolerarán un peor rendimiento mientras se recogen las estadísticas de las peticiones. En segundo lugar, asume que los patrones de acceso de los usuarios son invariantes con el tiempo. En el caso de que estas suposiciones no sean válidas, pueden conducir a disminuciones en el rendimiento que podrían no ser tolerables en función de las necesidades del servicio (Quinn y Gahegan, 2010).

Evidentemente, para experimentar una mejora por el uso de caché es necesario, en primer lugar, que ésta sea poblada con objetos. Existen dos procedimientos para poblar una caché de teselas:

1) Por una parte, el proceso mediante el cuál las teselas se generan y *cachean* de forma automática se conoce como *seeding*. La ventaja de este proceso es que mejora en gran medida la experiencia de usuario. El inconveniente es que se trata de un proceso que consume más tiempo y recursos de almacenamiento de los estrictamente necesarios (es posible que se pregeneren teselas que nunca sean pedidas).

2) La otra posibilidad, conocida como «cacheo bajo demanda», consiste en dejar que

[2] http://www.idee.es

las teselas sean generadas la primera vez que son pedidas, siendo cacheadas en ese mismo proceso. De esta forma, la primera vez que se pide una tesela se produce un fallo (*miss*) de caché y la petición se resuelve aproximadamente a la misma velocidad que si se tratase de un servicio WMS tradicional, mientras que las peticiones subsiguientes son aceleradas en gran medida dado que ya han sido generadas y se producen sucesivos aciertos (*hit*) de caché.

La principal ventaja de este último método es que no requiere pre-procesamiento y se produce una rudimentaria autogestión de la caché, puesto que las propias peticiones acumuladas en la memoria del sistema equivalen a una evaluación implícita de la probabilidad de petición de las teselas. El resultado indirecto es que sólo los datos pedidos son cacheados (los que efectivamente tienen una probabilidad no nula de ser pedidos), ahorrando por tanto recursos de almacenamiento. El inconveniente de este método es que la devolución de teselas no experimenta ninguna mejora de QoS en la primera petición, reduciendo la calidad en la experiencia del usuario en las teselas poco demandadas o ya expiradas.

Otro importante mecanismo que puede ayudar a mejorar el rendimiento de los servicios de mapas teselados es la carga dinámica mediante heurísticas. Ante una petición de un usuario, el sistema puede predecir cuál puede ser la siguiente petición o grupo de peticiones. Generando el resultado de estas predicciones dentro del intervalo de tiempo entre dos peticiones sucesivas se puede conseguir una mejora sustancial en la experiencia de usuario, a condición de que las predicciones sean acertadas.

Por otra parte, en el caso habitual de que no se disponga de espacio suficiente para poder albergar todos los objetos en la caché, cuando ésta se encuentra completa un algoritmo de reemplazo debe determinar qué teselas deben ser reemplazadas.

La principal problemática reside, por tanto, en las estrategias de estos sistemas de caché: precarga inicial (*seeding*), carga dinámica o heurística, y políticas de borrado o reemplazo.

Actualmente, la mayoría de estos algoritmos de gestión de caché se basan en principios extraídos de otros ámbitos, como las cachés en la jerarquía de memoria de los ordenadores (Tanenbaum, 2001) y las cachés Web como *Squid* (Wessels et al., 2009).

Sin embargo, muchos de ellos podrían explotar la naturaleza espacial de la información geográfica que manejan para optimizar su funcionamiento. Las características particulares de la información geográfica de una caché espacial permiten suponer que los algoritmos de reemplazo y carga inicial de una caché deben tener en cuenta las características espacio-temporales del comportamiento de los usuarios y la correlación multidimensional de las teselas.

Los sistemas de caché estudiados, como *TileCache* y *GeoWebCache* (OpenGeo, 2008; MetaCarta, 2008), utilizan algoritmos de reemplazo muy básicos (LRU - *Least Recently Used*) para la gestión de sus cachés que, en su uso en aplicaciones no geográficas, han demostrado no ser los más eficaces cuando los objetos no son homogéneos (Abrams et al., 1996).

Por otra parte, los sistemas de caché antes mencionados ofrecen mecanismos muy rudimentarios para la selección de regiones de actuación de las tareas de gestión de la caché. Por ejemplo, permiten indicar el encuadre o *bbox* (*Bounding Box*) del conjunto de teselas que se quiere cachear para las escalas indicadas. *TileCache* ofrece la posibilidad de cachear las teselas inscritas en la circunferencia cuyo centro y radio se especifican (MetaCarta, 2008). Sin embargo, estas posibilidades manuales se basan en la intuición del administrador. Una importante línea de investigación surge de la necesidad de disponer de mecanismos automáticos o semi-automáticos para la identificación avanzada de regiones potencialmente

candidatas a ser generadas durante el proceso de *seeding*.

En cuanto a los mecanismos de carga dinámica, también son muy rudimentarios. Por ejemplo, la implementación de caché *GeoWebCache* (Liu y Nie, 2010) incorpora una técnica, conocida como *metatiling*, que consiste en solicitar una imagen de mayor tamaño que la pedida por el cliente y que contenga también las teselas adyacentes para reducir el número de peticiones al servidor de mapas remoto. Por otra parte, el cliente de mapas OpenLayers aplica un *buffer* sobre la vista de mapa para precargar teselas adyacentes. En (Yesilmurat y Isler, 2012) se propone precargar también las teselas adyacentes a una solicitada.

1.1. Objetivos de la tesis y contribuciones

El objetivo de la presente tesis es la *propuesta y validación de nuevas estrategias de gestión de una caché de mapas que aprovechen la naturaleza espacial de los objetos que gestiona para mejorar el rendimiento de los servicios de mapas teselados en el contexto de las Infraestructuras de Datos Espaciales.*

Para poder lograr este objetivo se plantean una serie de **objetivos parciales** que permiten su consecución:

- Realizar un estudio del estado de arte, desde los servicios de mapas Web tradicionales, hasta su evolución hacia sistemas de mapas teselados. Se analizan tanto las soluciones estándar (principalmente las de OGC) como las soluciones propietarias (Google Maps, Microsoft Bing Maps, Nasa World Wind, etc.).
- Identificar sinergias con las estrategias de mantenimiento de caché existentes en otros ámbitos, como las cachés Web o las jerarquías de memoria de los ordenadores. Para ello se realizarán los siguientes estudios:
  - Estudio del estado de arte de las políticas de gestión de las cachés Web como Squid. Las cachés de mapas constituyen un caso específico de las cachés Web, con la particularidad de que éstas gestionan un tipo concreto de objetos, que son las teselas de mapa. Por tanto, es probable que gran parte del *know-how* adquirido en este ámbito pueda ser aplicado a este caso particular.
  - Estudio del estado de arte de los sistemas de jerarquía de memoria de los ordenadores. En este ámbito existe el concepto de localidad espacial entre los registros de memoria. Técnicas como la transferencia en bloques de memoria pueden ser fácilmente trasladados al contexto de los servicios de mapas. La técnica de *metatiling* es un buen ejemplo de ello.
- Evaluar la aplicabilidad de técnicas de aprendizaje automático (*machine learning*) en las estrategias de caché de mapas.
  - Estudio del estado de arte de estrategias de gestión de caché basadas en técnicas de aprendizaje automático, como redes neuronales (ElAarag y Romano, 2009b; Venketesh y Venkatesan, 2009; Romano y ElAarag, 2011), algoritmos genéticos (Vakali, 2002; Ali y Shamsuddin, 2009) o lógica difusa (Khajoueinejad et al., 2006; Ali y Shamsuddin, 2009).
  - Ampliar estos algoritmos para que tengan en cuenta la naturaleza espacial de los objetos o diseñar otros nuevos.

- Proponer estrategias automáticas o semi-automáticas de gestión y mantenimiento específicas de una caché de mapas: estrategias de población inicial ó *seeding*, de reemplazo y de precarga dinámica ante las peticiones de los usuarios.
- Evaluar las soluciones propuestas. Una vez desarrollados los algoritmos de gestión se implementarán en un sistema de caché, como GeoWebCache, para su validación en un entorno real.

El desarrollo de esta tesis doctoral ha dado lugar a las **contribuciones** resumidas a continuación:

- Estrategias de población inicial de la caché:
  - Modelo descriptivo para la precarga de teselas a partir de los registros de peticiones pasadas del servicio.
  - Modelo regresivo OLS para la identificación de regiones prioritarias a partir de un catálogo de fenómenos geográficos (o *features*) y un breve registro de accesos pasados.
  - Modelo predictivo basado en redes neuronales para el aprendizaje automático de regiones de interés a partir de un catálogo general de fenómenos geográficos y un breve registro de entrenamiento con peticiones pasadas.
- Estrategias de reemplazo de caché:
  - Estrategia de reemplazo de caché mediante redes neuronales.
  - Estrategia Perfect-LFU simplificada que aprovecha la correlación espacial de las peticiones.
- Estrategia de carga dinámica mediante *Metatiling* adaptativo.

El contexto, objetivos y contribuciones de la presente tesis se recogen de manera esquemática en la Figura 1.2.

1.2. Metodología

Para conseguir los objetivos propuestos en esta tesis se ha seguido la metodología de investigación definida en (Adrion, 1993), la cual establece que todo proceso de investigación en ingeniería debe seguir las cuatro etapas siguientes:

- Estudio de las soluciones de caché existentes. En esta primera etapa se realiza un estudio de estado de arte de las soluciones existentes con el fin de detectar puntos de mejora y problemas no solventados. En la literatura se encuentran escasas soluciones que tengan en cuenta la naturaleza espacial de los objetos para optimizar su rendimiento.
- Propuesta de una mejor solución. A partir de los estudios previos y teniendo en cuenta las particularidades de los servicios de caché de mapas, se proponen nuevos algoritmos de mantenimiento para estos servicios. Asimismo, se identifican aquellas soluciones técnicas existentes que se pueden aplicar al sistema de caché de mapas

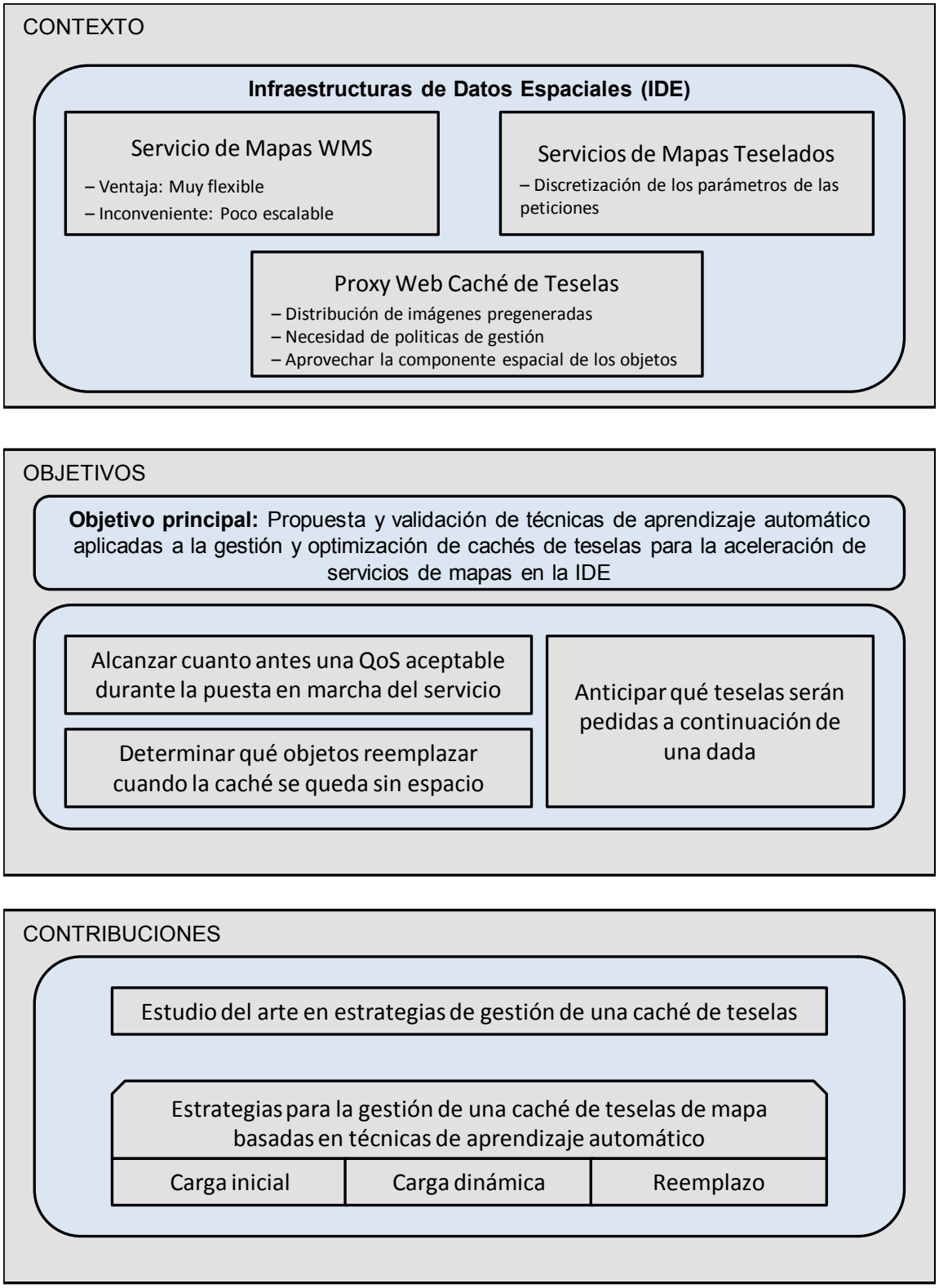


Figura 1.2: Contexto, objetivos y contribuciones de la tesis doctoral.

para proceder a su análisis más exhaustivo, seleccionando la técnica más adecuada, proponiendo y diseñando una mejor solución al problema, que supere las limitaciones detectadas en las existentes.

- Desarrollo de la nueva solución. En esta etapa se desarrolla la solución propuesta en la etapa anterior, dando lugar a un prototipo del sistema. La solución desarrollada será implementada sobre un producto *OpenSource* para que pueda ser aprovechada por la comunidad.

- Evaluación de la nueva solución. En esta última etapa se evalúa si el sistema propuesto cumple con los objetivos buscados, superando los problemas detectados, a partir de su aplicación en casos reales.

Esta metodología se aplica cíclicamente con el fin de mejorar y refinar progresivamente la solución presentada en la tesis.

1.3. Estructura de la memoria

En esta tesis se han cubierto todos los objetivos expuestos anteriormente, tal y como se describe a lo largo de esta memoria, cuya estructura general es la que se muestra a continuación.

Después de esta introducción, en el Capítulo 2 se comienza describiendo el actual servicio *Web Map Service* (WMS) como ejemplo prototípico de servicio de mapas convencional. A continuación se comentan las principales limitaciones de este servicio que motivaron la aparición de los servicios de mapas teselados, pasando posteriormente a describir las soluciones más relevantes basadas en esta técnica y los clientes que permiten acceder a dichos servicios. Estos servicios posibilitan la introducción de estrategias de caché, como se comenta en el Capítulo 3, donde se caracteriza formalmente la problemática inherente al teselado. En el Capítulo 4 se realiza un detallado estudio del arte en el que se recogen diversos trabajos encontrados en la literatura relacionados con la problemática de la gestión de una caché de teselas de mapa. En el Capítulo 5 se realiza un estudio del tráfico de peticiones de diversos servicios de mapas teselados, indicando las principales sinergias y divergencias con las características de los servicios Web convencionales. A continuación, en el Capítulo 6 se proponen dos novedosas estrategias de reemplazo de caché específicas para servicios de mapas; La primera de ellas se basa en el uso de redes neuronales, mientras que la segunda es una variante de la popular estrategia LFU (*Least Frequently Used*) que aprovecha la localidad espacial de los objetos. En el Capítulo 7 se presentan las dos nuevas estrategias de precarga propuestas para la población inicial de la caché; Una basada en un modelo descriptivo y otras dos en un modelo predictivo mediante la identificación de fenómenos geográficos directores de las peticiones de los usuarios. Por su parte, en el Capítulo 8 se propone una nueva estrategia de carga dinámica de teselas basada en el método de *metatiling* adaptativo. Finalmente se presentan las conclusiones y líneas futuras en el Capítulo 9. En el Apéndice A se describen diversos prototipos, desarrollados inicialmente para la experimentación con las diversas estrategias de gestión de una caché de mapas, liberados ahora bajo licencia *Open Source* para su uso por parte de la comunidad. En el Apéndice B se realiza un listado de los trabajos realizados en el contexto de esta tesis.

Capítulo 2

Servicios Web de Mapas

Resumen: Los Servicios Web de Mapas permiten el acceso a imágenes cartográficas a través de Internet. En este capítulo se comienza describiendo el actual servicio WMS de OGC como ejemplo prototípico de servicio de mapas convencional. A continuación se comentan las principales limitaciones de este servicio que motivaron la aparición de los servicios de mapas teselados, pasando posteriormente a describir las soluciones basadas en esta técnica más relevantes y los clientes que permiten acceder a dichos servicios.

2.1. Web Map Service (WMS)

El Servicio Web de Mapas (WMS) fue el primer servicio especificado por OGC (Allan Doyle, 2000) y puede considerarse como el más maduro por los siguientes motivos:

- Es el servicio OGC más implementado con diferencia, con 1127 implementaciones catalogadas en OGC-Services.NET (Skylab Mobilesystems Ltd, 2005).
- Ha sido adoptado como norma internacional ISO (ISO 19128:2005 *Geographic information: Web map service interface*) por el Comité Técnico ISO/TC 211 *Geographic information/Geomatics*[1] (ISO, 2009) en el año 2005.
- El laboratorio JPL (*Jet Propulsion Laboratory*) de la NASA dispone de un servicio WMS para compartir una gran variedad de datos geofísicos (Plesea, 2012).
- La mayor parte de aplicaciones SIG (Sistemas de Información Geográfica) del mercado soportan la visualización de servicios WMS. Además, muchos Globos Virtuales permiten mostrar la cartografía obtenida a partir de estos servicios.

Un Servicio Web de Mapas (WMS) produce mapas de datos referenciados espacialmente (o geo-referenciados) de forma dinámica a partir de información geográfica. Este estándar internacional define un «mapa» como una representación de información geográfica en forma de un archivo digital de imagen adecuado para ser mostrado en la pantalla de un ordenador. Los mapas producidos por un servicio WMS se generan normalmente en un formato de

[1] http://www.isotc211.org

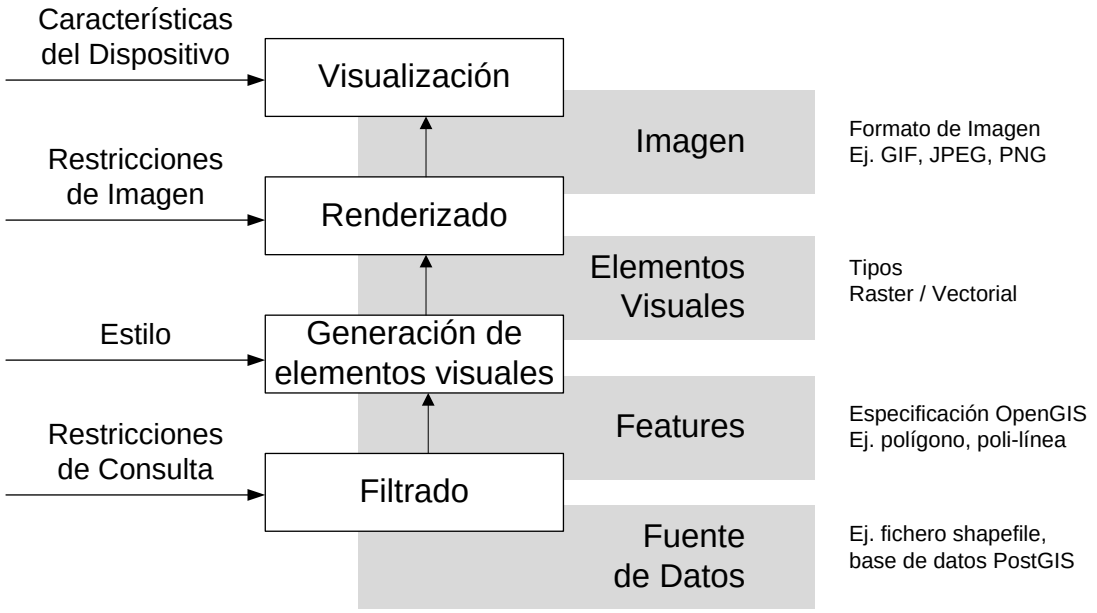


Figura 2.1: Modelo OpenGIS del *workflow* para la representación de una imagen de mapa. Imagen adaptada de (Doyle y Cuthbert, 1998).

imagen como PNG (*Portable Network Graphics*), GIF (*Graphics Interchange Format*) o JPEG (*Joint Photographic Experts Group*), o en ocasiones en formatos de imagen vectoriales como SVG (*Scalable Vector Graphics*) o WebCGM (*Web Computer Graphics Metafile*).

OGC describe la visualización de datos geoespaciales en forma de mapa como un *workflow* compuesto por cuatro procesos (Doyle y Cuthbert, 1998), tal y como se muestra en la Figura 2.1:

1. Selección de los datos geoespaciales que van a ser mostrados.
2. Generación de elementos de visualización a partir de los datos seleccionados.
3. Renderización de los elementos de visualización en forma de imagen.
4. Muestra de la imagen al usuario.

El estándar WMS define tres operaciones, dos de ellas (*GetCapabilities* y *GetMap*) obligatorias y otra (*GetFeatureInfo*) opcional:

- *GetCapabilities*: devuelve metadatos del nivel de servicio. Esta operación es genérica para todos los servicios OGC.
- *GetMap*: devuelve un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos.
- *GetFeatureInfo*: devuelve información acerca de características particulares mostradas en el mapa.

Tabla 2.1: Parámetros de una petición *GetMap*

Parámetro	Obligatorio / Opcional	Descripción
VERSION	Obligatorio	Versión del servicio
REQUEST	Obligatorio	Tipo de operación (*GetMap*)
LAYERS	Obligatorio	Lista separada por comas de una o más capas de mapa
STYLES	Obligatorio	Lista separada por comas con un estilo de renderizado por cada capa solicitada
CRS	Obligatorio	Sistema de referencia de coordenadas
BBOX	Obligatorio	Esquinas (inferior izquierda, superior derecha) [minx,miny,maxx,maxy] del *Bounding box* en las unidades del CRS
WIDTH	Obligatorio	Anchura en píxels de la imagen de mapa
HEIGHT	Obligatorio	Altura en píxels de la imagen de mapa
FORMAT	Obligatorio	Formato de salida de la imagen de mapa
TRANSPARENT	Opcional	Transparencia del fondo de la imagen [TRUE|FALSE] (default=FALSE)
BGCOLOR	Opcional	Valor RGB en Hexadecimal del color del fondo de la imagen (default=0xFFFFFF)
EXCEPTIONS	Opcional	El formato en que el WMS va a reportar las excepciones (default=XML)
TIME	Opcional	Valor de tiempo de la capa deseada
ELEVATION	Opcional	Elevación de la capa deseada
Other sample dimension(s)	Opcional	Valor para otras dimensiones cuando sea conveniente

Tabla 2.2: Petición WMS GetMap al servicio del PNOA que devuelve la imagen de mapa de la Figura 2.2.

```
http://www.idee.es/wms/PNOA/PNOA?LAYERS=PNOA&SERVICE=WMS&VERSION=1.1.
1&REQUEST=GetMap&STYLES=&EXCEPTIONS=&FORMAT=image/png&SRS=EPSG:4326&WIDTH=
768&HEIGHT=768&BBOX=-4.7072899365234,41.660670063476,-4.7031700634766,41.
664789936524
```

Tabla 2.3: Petición WMS GetMap al servicio del Catastro que devuelve la imagen de mapa de la Figura 2.3.

```
http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?LAYERS=
Catastro,CONSTRU,TXTCONSTRU,SUBPARCE,TXTSUBPARCE,PARCELA,TXTPARCELA,MASA,
TXTMASA,EJES,LIMITES,TEXTOS,ELEMLIN&FORMAT=image/png&SERVICE=WMS&VERSION=1.
1.1&REQUEST=GetMap&STYLES=&EXCEPTIONS=application/vnd.ogc.se_inimage&SRS=
EPSG:4326&BBOX=-4.7072899365234,41.660670063476,-4.7031700634766,41.
664789936524&WIDTH=768&HEIGHT=768
```

Una de las grandes ventajas de este servicio es que las operaciones anteriores pueden invocarse utilizando un navegador web estándar, realizando peticiones en forma de URL (*Uniform Resource Locator*), no siendo necesario para ello el uso de un cliente GIS. El contenido de tales URLs depende de la operación solicitada. En particular, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de referencia de coordenadas, y la anchura y la altura de la imagen de salida. En la Tabla 2.1 se muestran los parámetros, tanto obligatorios como opcionales, de una petición WMS *GetMap*.

Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados pueden solaparse para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (p.ej., GIF o PNG) o píxeles semi-transparentes permite que los mapas subyacentes sean visibles. Además, puede solicitarse mapas individuales de diversos servidores. El servicio WMS permite así la creación de una red distribuida de servidores de mapas, a partir de los cuales los clientes pueden construir mapas a medida.

A continuación se presentan a modo de ejemplo varias peticiones realizadas a un servicio de mapas WMS.

La petición recogida en la Tabla 2.2 devuelve una ortofoto del servicio de mapas del PNOA (Plan Nacional de Ortofotografía Aérea)[2] donde se visualiza la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT) de la Universidad de Valladolid (UVA). La imagen de mapa generada como respuesta por el servidor WMS se muestra en la Figura 2.2.

En la Tabla 2.3 se recoge una petición WMS en la que se solicitan múltiples capas de información catastral, con los mismos parámetros geográficos y tamaño de imagen de salida que la petición anterior. En la Figura 2.3 se muestra la imagen producida.

Por último, en la Figura 2.4 se muestra el resultado de combinar las imágenes de mapa producidas por las peticiones anteriores.

[2] http://www.ign.es/PNOA

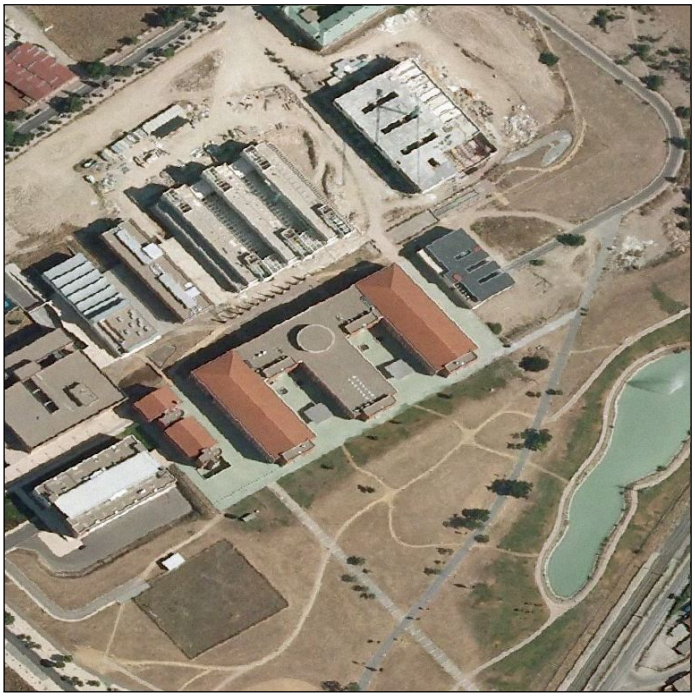


Figura 2.2: Ortofoto de la ETSIT de Valladolid obtenida a partir del servicio WMS del PNOA.

2.2. Servicios de Mapas Teselados

La especificación WMS fue desarrollada para servir mapas cartográficos de forma interoperable, y uno de sus principales objetivos fue la flexibilidad, permitiendo que los clientes obtuviesen exactamente la imagen final deseada. Así, un cliente WMS puede solicitar que el servidor cree un mapa superponiendo un número arbitrario de capas (*layers*) ofrecidas por el servidor, sobre una región geográfica arbitraria, con un color de fondo y a una escala arbitrarios, en cualquier sistema de referencia de coordenadas soportado. Asimismo, el cliente puede solicitar que las capas de mapa sean renderizadas utilizando un estilo específico anunciado por el servidor, o incluso usar un estilo proporcionado por el cliente cuando el servidor WMS implementa el estándar SLD (*Styled Layers Descriptor*) de OGC.

Sin embargo, debido a sus numerosos grados de libertad, los mapas necesitan generarse al vuelo. El dibujado de un mapa (ver Figura 2.1) es una tarea de gran carga computacional que dificulta que el servicio pueda responder adecuadamente a un número elevado de peticiones simultáneas. En general, los servicios WMS tienen serios problemas de escalabilidad.

Para afrontar este problema, los dominios continuos de los parámetros pueden reducirse a un conjunto discreto de valores. De esta forma, la zona geográfica queda dividida en una rejilla compuesta por elementos de geometría predefinida (denominados teselas - *tiles* en notación inglesa) e identificables mediante índices enteros. Esto posibilita la actuación de mecanismos de caché o incluso la prestación del servicio mediante una colección de imágenes pregeneradas. En éste último caso se reduce la carga computacional al mínimo mejorando el tiempo de respuesta y la escalabilidad.

Esta estrategia obliga a los clientes a realizar superposiciones de imágenes por sí mismos, en vez de ceder esta tarea al servidor. También limita a los clientes a solicitar imágenes

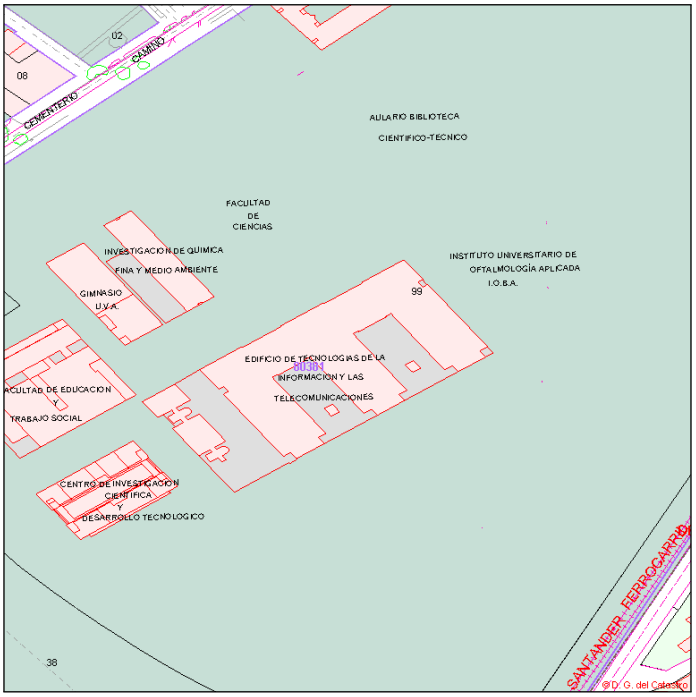


Figura 2.3: Imagen de mapa vectorial con cartografía catastral de la ETSIT y alrededores obtenida a través del servicio WMS de la Dirección General del Catastro Dir (2013).

Figura 2.4: Imagen de mapa resultante de combinar una ortofoto del PNOA con información catastral procedente de la Dirección General del Catastro.

de mapa que no están en la posición exacta deseada, forzando a los clientes a «pegar» varias teselas y «recortar» el resultado para formar la imagen final. Multitud de imple-

mentaciones comerciales como Google Maps[3], Microsoft Bing Maps[4] (antes conocido como Virtual Earth) y Yahoo! Maps[5] han demostrado que pueden conseguirse grandes mejoras de rendimiento adoptando esta estrategia.

Siguiendo esta estrategia, han surgido diversas soluciones no-propietarias abiertas e interoperables para la estandarización de los servicios de mapas teselados, entre las que pueden destacarse las especificaciones *WMS Tile Caching* (conocida como WMS-C) (Erle et al., 2006) y *Tile Map Service* de OSGeo (OSGeo, 2008a), y *Web Map Tile Service* (Masó et al., 2010) del OGC, de las cuales la primera es la más extendida.

2.2.1. Propuesta WMS Tile Caching (WMS-C) de OSGeo

La organización OSGeo ha publicado una estrategia de extensión de los metadatos de servicio del estándar WMS 1.1.1 que permite informar a los clientes de que el servidor ofrece un patrón concreto de teselado bien definido y disponible para su consulta. Esta estrategia se conoce como WMS-C.

La idea fundamental es que, a diferencia de WMS, dos peticiones distintas de una misma tesela deberían formar exactamente la misma petición HTTP GET. Esto motiva la necesidad de aplicar ciertas restricciones sobre las peticiones WMS GetMap:

1. Cadena mínima de argumentos de consulta (no se permiten argumentos opcionales).
2. Orden preestablecido de los argumentos y capitalización fija de los caracteres.
3. Rango fijo de posibles valores de *bounding box*.
4. Valores de *bounding box* con misma precisión decimal.
5. Tamaño fijo de las teselas expresado en pixels.
6. Nombres fijos de las capas y/o ordenación fija de los mismos.
7. Estilos fijos.
8. Formato de salida único.

Con esta implementación, la operación *GetCapabilities* del WMS incluye adicionalmente un elemento *VendorSpecificCapabilities* con la siguiente descripción DTD (OSGeo, 2008b):

```
<!DOCTYPE WMT_MS_Capabilities SYSTEM
"http://schemas.opengeospatial.net/wms/1.1.1/WMS_MS_Capabilities.dtd" [
<!ELEMENT VendorSpecificCapabilities (TileSet*) >
<!ELEMENT TileSet (SRS, BoundingBox?, Resolutions, Width, Height, Format, Layers*, Styles*) >
<!ELEMENT Resolutions (#PCDATA) >
<!ELEMENT Width (#PCDATA) >
<!ELEMENT Height (#PCDATA) >
<!ELEMENT Layers (#PCDATA) >
<!ELEMENT Styles (#PCDATA) >
]>
```

[3] http://maps.google.es
[4] http://www.bing.com/maps
[5] http://maps.yahoo.com

Los nuevos elementos informan a los clientes de la disponibilidad opcional de ciertas combinaciones de parámetros en forma de teselas *pregeneradas*. Dado que esta extensión afecta a un WMS estándar, las peticiones no enmarcadas dentro del espacio de teselas optativo deben seguir siendo respondidas con los resultados tradicionales de este servicio. De esta forma un servidor que siga la recomendación debería ser compatible con los estándares actuales y permitir un funcionamiento adaptado a la tecnología disponible en cada cliente.

Esta recomendación admite la inclusión en las peticiones de un parámetro opcional (*tiled*), que hace que el servidor se comporte como un servicio orientado exclusivamente a teselas (de esta forma se aparta de la especificación del OGC). Si se incluye en la petición el parámetro «tiled=true», los *bounding box* que no se ajusten al espacio de teselado definido en el *getCapabilities* serán respondidas con un error HTTP 404 o 500.

A continuación se reproduce la sección específica de la respuesta a una petición *GetCapabilities* a un servidor *WMS-C*:

```
<VendorSpecificCapabilities>
  <TileSet name="CARTOCIUDAD">
    <SRS>EPSG:4326</SRS>
    <BoundingBox SRS="EPSG:4326" minx="0" miny="0" maxx="22.5" maxy="22.5"/>
    <Resolutions>
      1.40625 0.703125 0.494384765625 0.3515625 0.17578125 0.087890625
      0.0439453125 0.0219726563 0.010986328125 0.0054931640625
    </Resolutions>
    <Width>256</Width>
    <Height>256</Height>
    <Format>png</Format>
    <Layers>
      Vial,Toponimo,SeccionCensal,FondoUrbano,CodigoPostal,Municipio
    </Layers>
    <Styles>default,style1</Styles>
  </TileSet>
</VendorSpecificCapabilities>
```

En el resto del documento de *Capabilities* se describen las capas ofertadas por el servidor en la modalidad estándar WMS.

La rejilla de referencia (denominada perfil) se define de forma explícita por medio del "*bounding box*" del servicio y el conjunto de resoluciones disponibles, tomando como origen de teselado la esquina inferior izquierda del "*bounding box*". Cualquier petición WMS cuyas coordenadas espaciales no se ajusten a este perfil generará un fallo de caché por la propia indefinición del dominio del servicio. En estos casos los servidores pueden devolver un error de servicio (obligatorio si se incluye el parámetro "*tiled=true*") o intentar trasladar la petición a los servicios WMS originales.

2.2.2. Web Map Tile Service (WMTS)

A pesar de que la propuesta WMS-C de OSGeo ha sido la solución teselada no-propietaria más adoptada hasta el momento, con la reciente adopción como estándar del servicio *Web Map Tile Service* (WMTS) (Masó et al., 2010) de OGC resulta previsible una mayor proliferación de este servicio.

La interfaz WMTS especifica tres operaciones que pueden ser solicitadas por un cliente: dos de ellas, *GetCapabilities* y *GetFeatureInfo*, presentes en el estándar WMS, y una nueva petición *GetTile* que sustituye a la petición *GetMap* del anterior. Estas peticiones pueden codificarse en KVP (*Key Value Pair*), REST y SOAP.

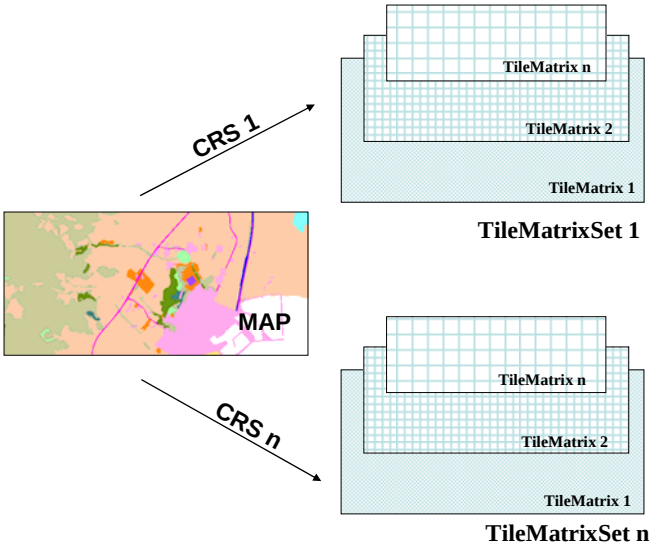


Figura 2.5: Estructura piramidal de teselas para una capa expuesta a través de un servicio WMTS. Fuente: (Masó et al., 2008).

Cada una de las capas (*layers*) de un servidor WMTS sigue una estructura piramidal de escalas, en la que cada escala o nivel de la pirámide es una rasterización y fragmentación de los datos geográficos a una escala o tamaño de píxel concreto (Masó et al., 2008).

Esta discretización del espacio se describe en WMTS a través de *TileMatrixSets*. Un *TileMatrixSet* consta de una o más «matrices de teselas» (*TileMatrix*) que definen las teselas disponibles en el servidor para cada escala y para un sistema de referencia de coordenadas concreto. Esta organización se ilustra en la Figura 2.5.

Cada una de estas matrices de teselas define los siguientes parámetros:

- Escala de resolución de las teselas expresado en forma de denominador de escala. Este denominador de escala se define en base a un tamaño de píxel estandarizado de 0.28 mm x 0.28 mm.
- Anchura y altura de cada tesela en píxels.
- Coordenadas de la esquina superior izquierda del *bounding box* de la matriz de teselas.
- Anchura y altura de la matriz en unidades de teselas.

El número de *TileMatrixSets* servidos por un servidor WMTS para una determinada capa en la que se ofrecen $nTileMatrices$ escalas, $nStyles$ posibles estilos y $nFormats$ posibles formatos de salida es de:

$$nTileMatrixSets = nTileMatrices \times nStyles \times nFormats \tag{2.1}$$

si no se definen otras dimensiones (p.ej., tiempo, altura, etc.), o en el caso de que se definan $nDimensions$ dimensiones:

$$nTileMatrixSets = nTileMatrices \times nStyles \times nFormats \times nDimensions \tag{2.2}$$

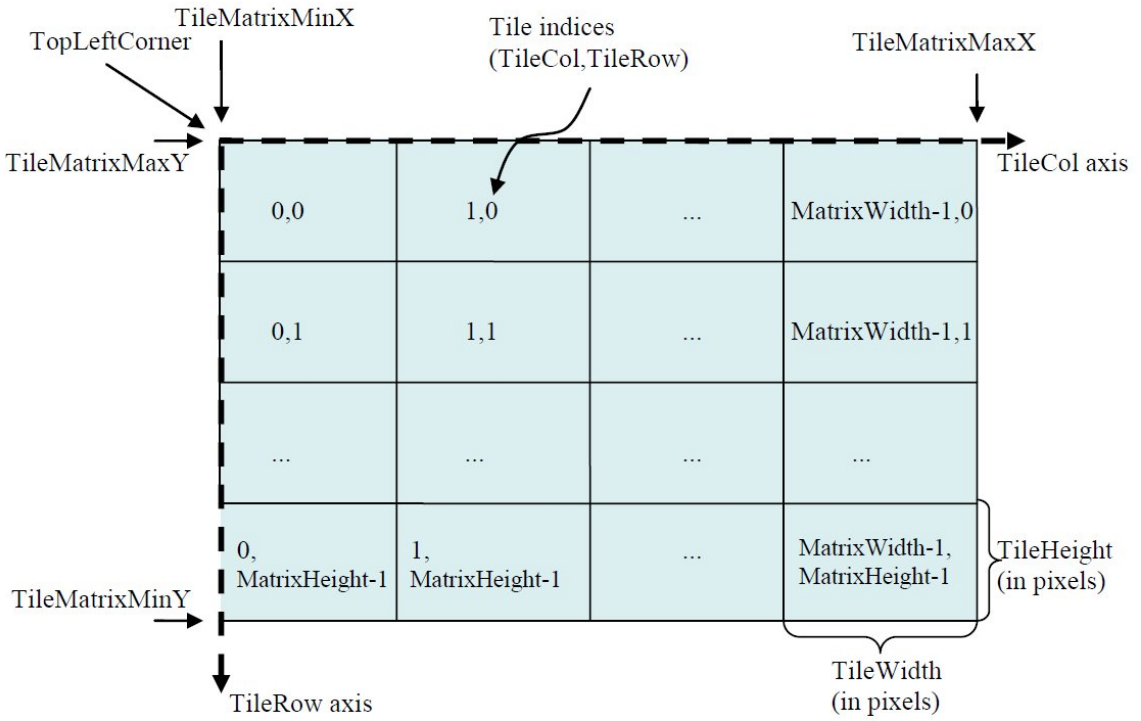


Figura 2.6: Espacio de teselado definido por el estándar WMTS. Fuente: (Masó et al., 2010).

Cada capa referencia a uno o más *TileMatrixSets*, puediendo haber múltiples capas que referencian un mismo *TileMatrixSet*.

Un *TileMatrixSet* tiene un *bounding box* aproximado (opcional) pero cada *TileMatrix* que lo compone tiene un *bounding box* exacto que se deduce de forma indireta a partir de otros parámetros. Los *bounding boxes* de cada *TileMatrix* a cada escala de resolución pueden variar ligeramente debido al alineamiento de los pixels, y es importante tener en cuenta esta variación.

Dadas las coordenadas de la esquina superior izquierda de un *TileMatrix* ($tileMatrixMinX$, $tileMatrixMaxY$), la anchura y altura de la matriz de teselas en unidades de teselas ($matrixWidth$, $matrixHeight$), la anchura y altura en pixels de cada tesela ($tileWidth$, $tileHeight$), el coeficiente para convertir unidades del sistema de referencia de coordenadas (CRS (*Coordinate Reference System*)) a metros (metersPerUnit) y la escala (1:scaleDenominator), la esquina inferior derecha del *bounding box* de un *TileMatrix* ($tileMatrixMaxX$, $tileMatrixMinY$) puede calcularse de la forma siguiente (Masó et al., 2010):

$$pixelSpan = scaleDenominator \times 0{,}2810^{-3}/metersPerUnit(crs) \tag{2.3}$$

$$tileSpanX = tileWidth \times pixelSpan \tag{2.4}$$

$$tileSpanY = tileHeight \times pixelSpan \tag{2.5}$$

$$tileMatrixMaxX = tileMatrixMinX + tileSpanX \times matrixWidth \tag{2.6}$$

$$tileMatrixMinY = tileMatrixMaxY - tileSpanY \times matrixHeight \quad (2.7)$$

En la Figura 2.6 se muestra el espacio de teselas definido por un *TileMatrix*.

2.2.3. Microsoft Bing Maps y Google Maps

Al margen de las especificaciones y estándares anteriores, Microsoft ha desarrollado su propio esquema de teselado para su servicio de mapas Bing Maps[6]. Este servicio utiliza una proyección cilíndrica Mercator, ilustrada en la Figura 2.7. En este servicio, el mapa se divide en teselas de tamaño fijo 256x256 píxels. El nivel de detalle (LOD - *Level of Detail*) más bajo contiene 4 teselas y cada nivel de detalle duplica el número de teselas en cada dirección (cuatro veces más en total) que el inmediatamente inferior. De esta forma, el número de teselas en el nivel de resolución k es $n = 4^k$. Cada tesela se indexa con un par de coordenadas (X, Y) que van desde la $(0, 0)$ en la esquina inferior izquierda, hasta la $(2^k - 1, 2^k - 1)$ en la esquina inferior derecha (ver Figura 2.8).

Figura 2.7: Proyección Mercator utilizada por Bing Maps.

Para optimizar el indexado y almacenamiento de las teselas, las coordenadas XY de una tesela se combinan en una cadena uni-dimensional denominada *quadkey*. Cada *quadkey* identifica una tesela en un determinado nivel de resolución. Para convertir de coordenadas XY a *quadkey* hay que intercalar los bits de ambas coordenadas, interpretando el resultado como un número expresado en base 4. Por ejemplo, dada una tesela con coordenadas XY de (3, 5) en el nivel de resolución 3, el *quadkey* correspondiente se obtiene de la forma siguiente:

X = 3 = $\mathbf{011}_2$

Y = 5 = $\underline{101}_2$

quadkey = $\underline{\mathbf{1}}\mathbf{0}\underline{0}\mathbf{1}\underline{\mathbf{1}}\mathbf{1}_2 = 213_4$ = "213"

[6] http://www.bing.com/maps/

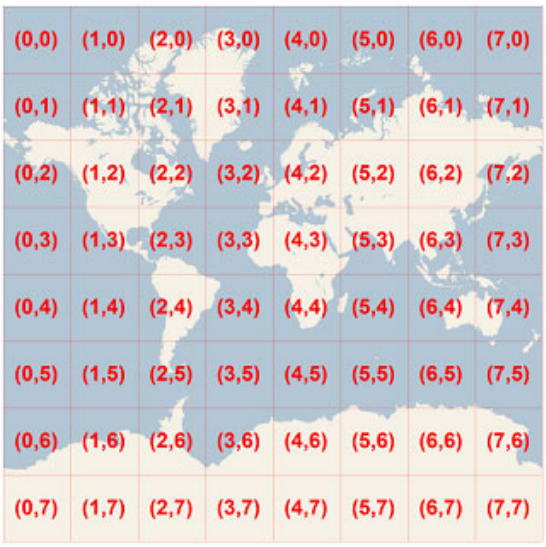


Figura 2.8: Esquema de teselado en Bing Maps para la escala 3.

Los *quadkeys* presentan una serie de propiedades muy interesantes. Por una parte, la longitud de un *quadkey* (número de digitos) corresponde al nivel de resolución de la tesela correspondiente. Por otra parte, el *quadkey* de una tesela comienza con aquel correspondiente de la tesela que lo contiene en el nivel de resolución inmediatamente anterior. Todo ello se aprecia en la Figura 2.9 (Schwartz, 2009).

El esquema de teselado definido por Google para su herramienta Google Maps es muy similar al de Microsoft antes comentado. Utiliza un sistema de coordenadas propietario denominado EPSG:900913 ("900913" corresponde a la palabra "GOOGLE" en alfabeto 1337), muy parecido al EPSG:3857[7].

2.3. Clientes de Mapas

Hoy en día, existen multitud de clientes de mapas que ofrecen la posibilidad de acceder a servicios de mapas teselados para obtener la cartografía a representar. Con la aparición de la tecnologia AJAX (*Asynchronous JavaScript and XML*) se popularizaron, frente a las tradicionales soluciones de escritorio, los clientes de mapas *Web*, como *Google Maps*, *Yahoo Maps*, *Microsoft Bing Maps* u *Openlayers*, por citar los más populares.

El uso de AJAX evita el refresco o «parpadeo» de la página que se producía al arrastrar el mapa en los clientes Web tradicionales (Mahemoff, 2006). En estos últimos, sin soporte para la petición de teselas, un ligero desplazamiento del mapa obligaba a solicitar una nueva imagen completa para el nuevo encuadre del mismo, a pesar de que la imagen era la misma prácticamente en su totalidad.

Con la adopción de la tecnología AJAX y los clientes teselados, un pequeño desplazamiento del mapa únicamente desencadena la petición de unas pocas teselas para cubrir la porción del mapa entrante fruto de este desplazamiento. De esta forma, el cliente no tiene que esperar un largo tiempo hasta que se actualice el mapa en la pantalla.

[7] http://spatialreference.org/ref/sr-org/6864/

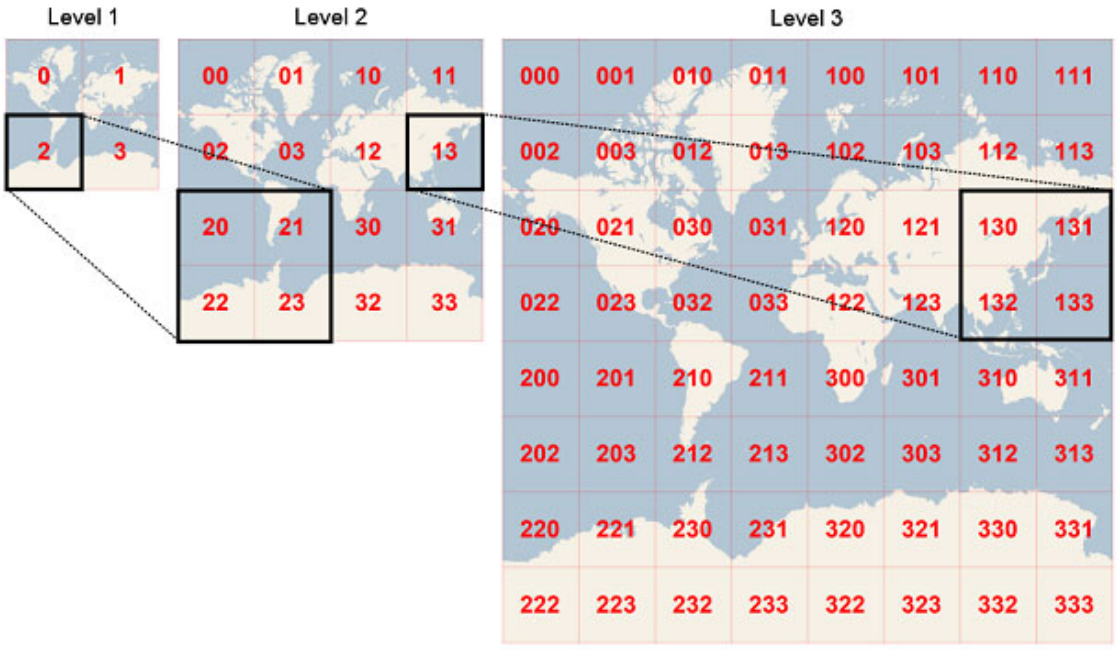


Figura 2.9: *Quadkeys* en Bing Maps.

Para mejorar aún más la experiencia del usuario, algunos clientes de mapas, como *Openlayers*, precargan las teselas adyacentes a las estrictamente necesarias para el área a visualizar. De esta forma, al desplazarse por el mapa, las nuevas teselas aparecen sin retardo alguno ofreciendo una mayor sensación de continuidad. En la Figura 2.10 se muestran las teselas incluidas alrededor de la zona visible del mapa para distintos valores de *buffer*.

Además, algunos clientes, como los globos virtuales *Nasa World Wind* o *Google Earth*, disponen de una caché en la que almacenan las teselas devueltas por el servidor de mapas remoto, de forma que si éstas son pedidas posteriormente pueden ser devueltas a partir de la caché más rápidamente.

Dada la gran heterogeneidad de API existentes, sería deseable disponer de un nivel mayor de abstracción que permitiese al desarrollador de *mashups* poder cambiar de forma rápida y sencilla de cliente de mapas. Esta fue la motivación para la aparición de Mapstraction[8], una librería Javascript que provee un API común para diversos proveedores de mapas 2D. Este tipo de API (*Application Programming Interface*) recibe el nombre de «API Universal y Políglota».

Debido a las importantes carencias encontradas en esta librería, durante el transcurso de la presente investigación se ha desarrollado la extensión IDELabMapstraction. Esta extensión aporta importantes mejoras a la librería original, como la mejora en la interactividad del usuario con los componentes del mapa, el soporte para recomendaciones OGC, posibilidad de integración como un componente GWT (*Google Web Toolkit*) o la incorporación de soporte específico para globos virtuales (López et al., 2010b, 2011a; García et al., 2011h,a).

Por otra parte, la madurez de determinadas tecnologías y el cambio del perfil de los usuarios de consumidor a generador de contenidos han popularizado el uso de CMS (*Con-*

[8] http://www.mapstraction.com

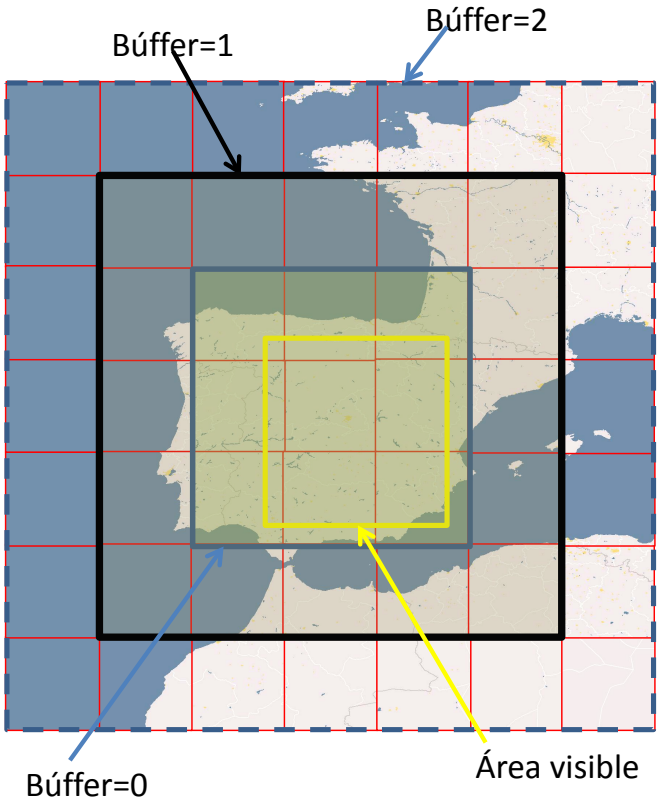


Figura 2.10: Teselas cargadas por el cliente OpenLayers para la región visualizada con distintos valores de *búffer* de pre-carga.

tent Management System). Se trata de plataformas que proporcionan la base para poder generar webs colaborativas de forma sencilla y sin necesidad de tener excesivos conocimientos previos. Una de las posibilidades que todavía no se han explotado suficientemente en estos sistemas es la georreferenciación de contenidos. De esta forma, aparece una nueva categoría de enlaces semánticos en base a las relaciones espaciales. En el actual estado de la técnica, se puede aprovechar la potencia de las bases de datos espaciales para manejar contenidos georreferenciados y sus relaciones espaciales, pero prácticamente ningún CMS lo aprovecha. Por ello, se ha desarrollado un módulo para el CMS Drupal que proporciona un soporte verdaderamente espacial y una interfaz gráfica en forma de mapa, mediante las que se pueden georreferenciar los contenidos (López et al., 2010a).

Capítulo 3

Gestión y optimización de las cachés de servicios OGC teselados

Resumen: Los servicios de mapas teselados posibilitan la introducción de estrategias de caché mediante la discretización de los dominios de los parámetros de las peticiones. En este capítulo se caracteriza formalmente la problemática inherente al teselado. Asimismo, se establecen una serie de condiciones de contorno que acotan el escenario de aplicabilidad de las estrategias de gestión de la caché propuestas en los capítulos posteriores.

Históricamente, la solución a la escalabilidad de los servicios en Internet ha pasado por la utilización de sistemas de caché en diversos niveles (Hassanein et al., 2002) del sistema completo cliente/red y dentro de los servidores en las distintas interfaces de intercambio de entidades, según recomiendan las distintas arquitecturas de diseño y de despliegue. En (Shan y Hua, 2009) se muestra un interesante diagrama taxonómico (ver Figura 3.1) de los distintos patrones de caché clasificados por capas de arquitectura y por tecnologías típicas de aplicación.

En general, las operaciones de caché se basan en la existencia de un conjunto finito, o al menos numerable, de elementos identificables en un almacén de datos o sistema de recuperación. Esta característica permite establecer sistemas de almacenamiento rápido para albergar los elementos con más probabilidad de ser accedidos en un futuro próximo.

En el caso de los servicios de información geográfica, las recomendaciones del OGC establecen las interfaces de comunicación de los diferentes servicios buscando cubrir un amplio conjunto de casos de uso. La flexibilidad y modularidad de estas interfaces hacen que los sistemas implantados deban responder en ocasiones a peticiones de servicio en tiempo real muy exigentes que afectan negativamente a la QoS percibida por el usuario. Un ejemplo prototípico de esta situación es el servicio WMS (ver Capítulo 2.1) al que se le pueden realizar peticiones de mapas con los siguientes grados de libertad:

- Capa: todas las capas ofrecidas por el servicio pueden ser solicitadas en cualquier orden de superposición.
- Estilo: cada capa puede solicitarse con cualquiera de los estilos con nombre (*named style*) y, opcionalmente, con cualquier estilo proporcionado por el usuario en cada petición.

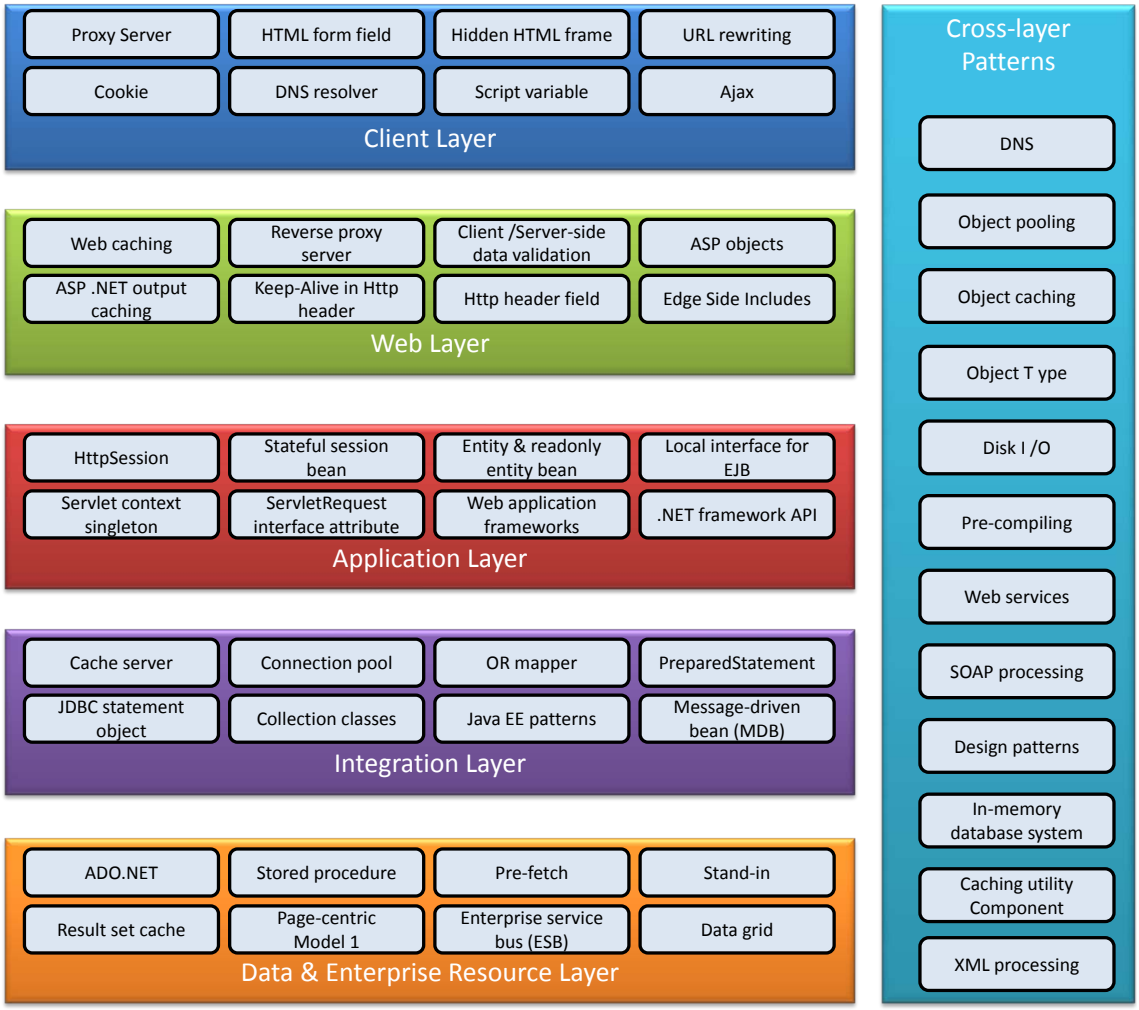


Figura 3.1: Taxonomía de los sistemas y puntos de actuación de las caché. Imagen adaptada de (Shan y Hua, 2009).

- Tamaño de imagen: Cualquier tamaño expresado en píxeles que el usuario necesite para su aplicación.
- Sistema de proyección: El mapa se puede pedir proyectado en cualquiera de los sistemas de proyección ofertados por el servidor.
- Área geográfica: Cualquier zona geográfica expresada mediante dos pares de coordenadas decimales de precisión arbitraria sin ninguna restricción en su dominio de definición.

Como al menos dos de los elementos anteriores (tamaño y área geográfica) tienen un dominio de definición continuo, el espacio de claves generado contiene un número infinito de elementos. Tal conjunto no puede beneficiarse de un sistema caché dado que la probabilidad efectiva de que una petición se repita es virtualmente nula. Tal y como se comentó en el Capítulo 2.2, para permitir el funcionamiento de los *proxy* o los *Web-Cache* se han

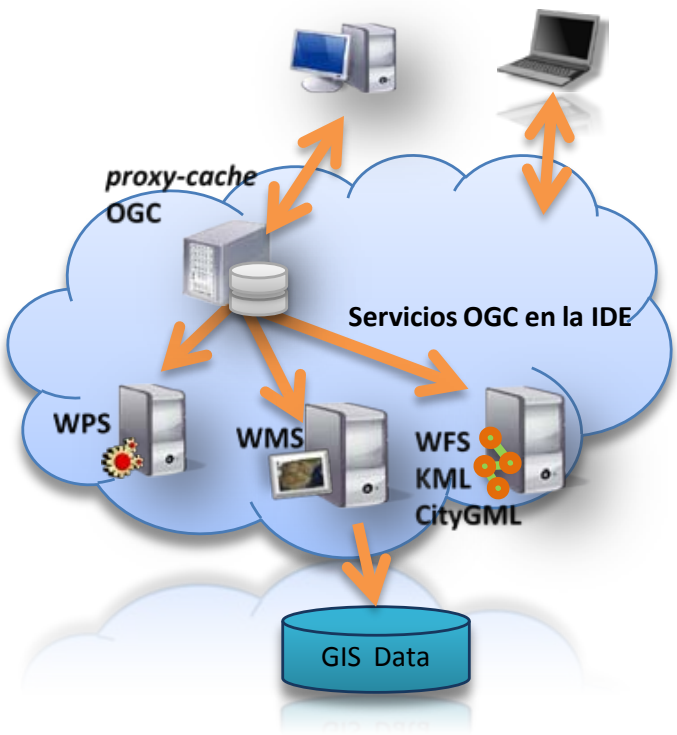


Figura 3.2: Relación general de un filtro tipo *Web Cache* con los servicios OGC generadores de objetos a partir de parámetros espaciales discretizables.

propuesto diversas recomendaciones para servicios de mapas teselados que se fundamentan en la acotación de los dominios de las variables de las peticiones. De esta forma todos los valores pertenecen a conjuntos discretos y la caché generará aciertos con probabilidad no nula. En general podemos decir que los sistemas de caché espacial son de aplicación para todas aquellas fuentes de información primarias cuyas primitivas de servicio puedan normalizarse y discretizarse al menos para un porcentaje significativo de peticiones de servicio. Además, cuando los servicios incorporan parámetros espaciales, es posible que existan correlaciones espaciales significativas entre las peticiones que puedan hacer más eficaces las técnicas de caché al permitir definir zonas de características o requisitos similares y adoptar en esas zonas técnicas específicas de cacheo.

Potencialmente, el campo de aplicación de estas tecnologías abarca a todos los servicios OGC dado que todos tienen una componente geográfica significativa. De hecho las conclusiones obtenidas de esta línea de investigación podrán ser de aplicación al resto de servicios de base espacial (ver Figura 3.2).

Durante el estudio del estado del arte se constató que las implementaciones comerciales y *Open Source* utilizan estrategias de mejora de la latencia pregenerando zonas de la cartografía de acuerdo a unos estilos de representación fijos. Generalmente, la información incluida en la caché es muy estática, por lo que es habitual disponer de una caché completa (conteniendo el 100 % de los objetos del dominio). Esta solución implica asumir dos problemas inmediatos relacionados con el consumo de recursos y con la calidad del servicio:

- Grandes requisitos de almacenamiento: Incluso para aplicaciones de escala modesta se

genera unos grandes requisitos de almacenamiento. En muchos casos hay que recurrir a cachés incompletas que requieren una adecuada política de gestión y mantenimiento.

- Retardo de la puesta en marcha: El tiempo de puesta en servicio aumenta con el tamaño de la zona geográfica de servicio y de la colección de escalas de representación. Durante un cierto periodo de tiempo (fase de QoS transitoria) la calidad del servicio se verá degradada hasta alcanzar asintóticamente los parámetros definitivos (fase de QoS estacionaria). Si la información debe actualizarse periódicamente, sería posible que no se alcanzase nunca la QoS objetivo.

La solución más general es, por lo tanto, considerar para el análisis un sistema de caché con recursos de almacenamiento y computacionales limitados y con unos objetivos de QoS que se deben obtener cuanto antes y mantener durante la vida del servicio. Los objetos a cachear tendrán un tiempo de vida tras el cual se consideran obsoletos y son descartados. Se supondrá que el sistema inicia su funcionamiento desde un estado vacío. Estas condiciones de contorno permitirán aplicar los resultados a sistemas de todas las escalas de despliegue y a diversas variedades de servicio basadas en parámetros espaciales.

Con estas premisas puede definirse formalmente una nomenclatura y una serie de conceptos para poder establecer métricas cuantitativas y poder aplicar criterios para discriminar metodologías de gestión y optimización de las cachés espaciales. En esta investigación se han analizado dos sencillos indicadores de la calidad de un servicio caché:

- la latencia τ; entendida como el tiempo necesario para la obtención de los objetos resultado e iniciar su transmisión.

- el tiempo de servicio $t_{service}$; percepción del consumidor del servicio de la espera necesaria hasta la finalización de la entrega del objeto solicitado. Este parámetro está compuesto de la latencia del servicio, la transferencia de la información y el procesado en el cliente. No se están considerando todos los efectos relacionados con las infraestructuras de red y el tamaño de los objetos.

En cualquier instante se puede definir un estado para una caché en la que los objetos que gestiona pueden estar disponibles con una cierta probabilidad.

Como los objetos de una caché espacial se pueden identificar por sus coordenadas, denominaremos a las teselas como $T(i,j,n)$, con $i,j,n \in \mathbb{N}$ (donde n es el nivel en la pirámide de escalas e i,j son los índices que sitúan dicha tesela en el plano n). Asimismo, denominaremos $P_h\{T(i,j,n)\}$ a la probabilidad de conseguir un acierto de caché al solicitar la tesela $T(i,j,n)$. Denominemos τ_h (hit) al coste en segundos necesario para obtener un objeto de la caché y τ_m $(miss)$ al coste incurrido al construir un nuevo objeto a partir de los servicios originales.

Sea $f_{req}(x,y,n)$ la densidad espacial de probabilidad que caracteriza la distribución espacial de los centroides de las peticiones $(requests)$ coincidentes con la escala n. En general, para cualquier distribución de las peticiones, las teselas son solicitadas con probabilidad (3.1).

$$P_{req}\{T(i,j,n)\} = \int_{y=y_{n,j}}^{y_{n,j+1}} \int_{x=x_{n,i}}^{x_{n,i+1}} f_{req}(x,y,n)\,dxdy \tag{3.1}$$

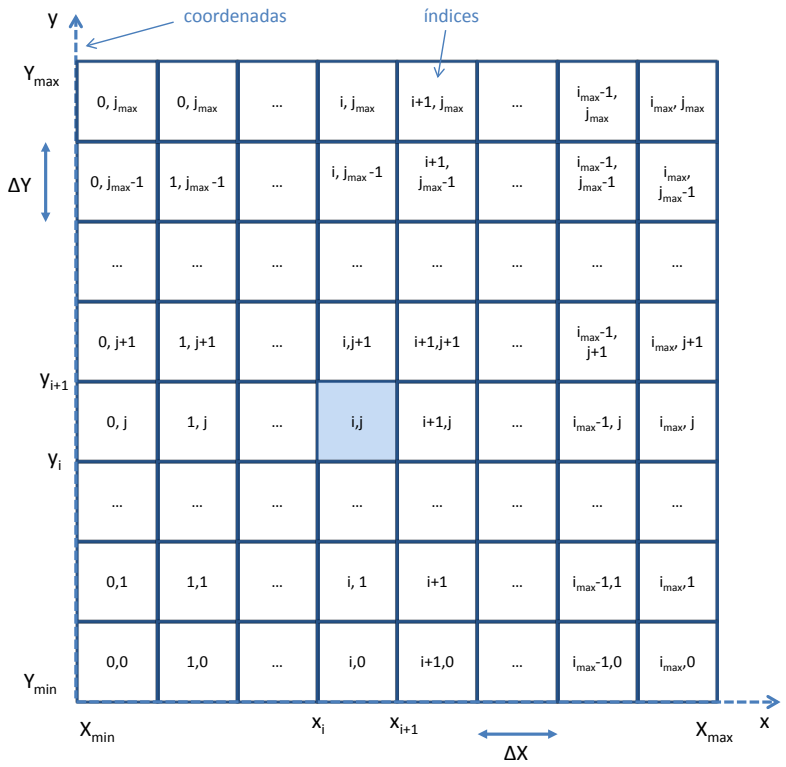


Figura 3.3: Espacio uniforme de teselado en un nivel concreto de la pirámide de escalas.

Aunque este resultado general nos será de utilidad a la hora de analizar las peticiones no teseladas dirigidas a un *proxy-cache*, en los análisis iniciales podemos simplificarla suponiendo una distribución uniforme de los centroides de las peticiones dentro de cada tesela, o bien que las peticiones están restringidas a la rejilla de referencia (ver Figura 3.3) como en el caso de un WMS-C. En estas condiciones la probabilidad de acceso a una tesela de coordenadas $T(i,j,n)$ y de tamaño $\Delta x \Delta y$ es

$$P_{req}\left\{T\left(i,j,n\right),t\right\} = f_{req}\left(x,y,n,t\right)\Delta x \Delta y \tag{3.2}$$

Para una petición individual de una tesela situada en la coordenada (discretizada) (i,j,n), en un instante t, la latencia observada viene determinada por (3.3):

$$\tau\left(i,j,n,t\right) = P_h\left\{T\left(i,j,n\right)\right\}\left(t\right)\tau_h + \left(1 - P_h\left\{T\left(i,j,n\right)\right\}\left(t\right)\tau_m\right) \tag{3.3}$$

Combinando las definiciones de (3.3) y (3.1) obtenemos una expresión probabilística conjunta de las peticiones que nos permite calcular una latencia media del servicio:

$$\tau\left(t\right) = \sum_{\langle ijn\rangle}\left(\tau_m - P_h\left\{T\left(i,j,n\right)\right\}\left(t\right)\left(\tau_m - \tau_h\right)\right)P_{req}\left\{T\left(i,j,n\right)\right\} \tag{3.4}$$

O equivalentemente

$$\tau\left(t\right) = \tau_h \sum_{\langle ijn\rangle}\left(\frac{\tau_m}{\tau_h} - P_h\left\{T\left(i,j,n\right)\right\}\left(t\right)\left(\frac{\tau_m}{\tau_h} - 1\right)\right)P_{req}\left\{T\left(i,j,n\right),t\right\} \tag{3.5}$$

Si suponemos que el tiempo de acceso a la caché es constante y que también lo es el tiempo necesario para generar nuevos objetos (al menos en término medio), podemos definir la ganancia de uso de caché como el incremento porcentual de rendimiento en los aciertos de caché:

$$G_c = \frac{\tau_m}{\tau_h} \tag{3.6}$$

y podemos reducir (3.4) a

$$\tau(t) = \tau_h \sum_{\langle ijn \rangle} \left(G_c - P_h\left\{T(i,j,n)\right\}(t)\left(G_c - 1\right)\right) P_{req}\left\{T(i,j,n), t\right\} \tag{3.7}$$

Donde ya se pueden identificar algunos componentes que deben ser caracterizados al menos localizadamente. En este punto no debemos asumir más simplificaciones probabilísticas que si bien podrían simplificar el diseño y funcionamiento de estos sistemas, eliminarían la información disponible para mejorar su gestión[1]. Por lo tanto parece necesario estudiar cuáles son las características típicas de estas propiedades y cómo se relacionan con las métricas de calidad.

Algunas características relevantes del modelo planteado son:

- No hay independencia estadística entre $P_h\left\{T(i,j,n)\right\}(t)$ y $P_{req}\left\{T(i,j,n), t\right\}$ ya que resulta evidente que el estado de la caché está vinculado íntimamente a la historia de las peticiones de servicio.
- No hay invarianza temporal durante el régimen transitorio del sistema.
- La función densidad de probabilidad (y por ende la probabilidad expresada en (3.1)) no es uniforme y seguramente presente vínculos directos con la estructura espacial de la información subyacente.

Otro factor a tener en cuenta es la magnitud práctica que supone gestionar estructuras de datos de dimensiones exponenciales. En una caché con estructura de *pirámide* de escalas, el número de elementos aumenta exponencialmente con la coordenada n por lo que la aplicación de algoritmos analíticos y predictivos se puede tornar impráctica, incluso mediante el apoyo de mecanismos heurísticos, si pretende aplicarse a todos los niveles de la pirámide.

Por este motivo resulta de gran interés obtener un modelo de relación estadística entre los distintos niveles de representación de la *pirámide* en el que se puedan extrapolar características cuantitativas de los niveles inferiores a partir de medidas recopiladas en un cierto nivel intermedio de la misma.

Vemos en la expresión (3.8) una estimación de la probabilidad de recibir peticiones en la zona geográfica de una tesela $T(i,j,n)$ calculada a partir de los niveles inferiores, hasta un cierto nivel L, según (3.1) (esta operación se ilustra en la Figura 3.4).

$$\hat{P}_{req}\left\{T(i,j,n)\right\} = \sum_{l=n}^{L} \int_{y=y_{n,j}}^{y_{n,j+1}} \int_{x=x_{n.i}}^{x_{n,i+1}} f_{req}(x,y,l)\, dxdy \tag{3.8}$$

Bajo la suposición de que los centroides de las peticiones presentan una distribución uniforme dentro de cada tesela, o bien que las peticiones están restringidas a la rejilla de

[1] Por ejemplo, suponer una caché completa reduciría (3.7) a $\tau(t) = \tau_m / G_c = \tau_h$

referencia (como en el caso de un WMS-C) puede hacerse uso de la expresión (3.2) para obtener la estimación de probabilidad (3.9).

$$\hat{P}_{req}\left\{T\left(i,j,n\right)\right\} = \sum_{l=n}^{L} f_{req}\left(x,y,n\right)\Delta x_n \Delta y_n \tag{3.9}$$

Para incrementar la interoperabilidad entre clientes y servidores se recomienda que las capas publicadas por estos últimos utilicen un conjunto concreto de escalas en el mismo sistema de referencia de coordenadas que la comunidad de usuarios acuerda utilizar. De esta forma, un cliente de mapas puede superponer teselas procedentes de distintos servidores sin necesidad de realizar reproyecciones ni re-escalado de teselas. Para ello se define el concepto de *Well-Known Scale Set* o «Conjunto de escalas bien conocidas», como una combinación de CRS y conjunto de escalas (Masó et al., 2010).

Los esquemas de teselado definidos por varios de estos *Well-Known Scale Set*, como el `GoogleCRS84Quad` o el `GoogleMapsCompatible` (Masó et al., 2010), definen una estructura piramidal en la que cada nivel de la pirámide representa la Tierra utilizando el doble de teselas en cada dirección que el nivel anterior (de resolución más gruesa). En este caso se tiene que $\Delta x_n = 2^k \Delta x_{n-k}$ y $\Delta y_n = 2^k \Delta y_{n-k}$, y la expresión (3.8) puede reducirse a

$$\hat{P}_{req}\left\{T\left(i,j,n\right)\right\} = \sum_{l=n}^{L} \sum_{\gamma=2^{l-n}i}^{2^{l-n+1}i-1} \sum_{\nu=2^{l-n}j}^{2^{l-n+1}j-1} P_{req}\left\{T\left(\gamma,\nu,l\right)\right\} \tag{3.10}$$

La validez de (3.8)-(3.10) parte de la hipótesis de que la localización geográfica es una propiedad relevante para todos los niveles (o al menos para niveles próximos) y que las peticiones tienen una correlación espacial elevada. De ser así, se podrían utilizar métodos heurísticos basados en el *principio de localidad* (Denning, 2005) para gestionar la totalidad de la caché utilizando *sondas* estadísticas en niveles con un dominio de objetos (*teselas*) abordable.

También resulta prometedor el análisis temporal a corto plazo. Si aparecen patrones de correlación espacio-temporal en las peticiones de los usuarios se pueden implementar operaciones de precarga predictiva que aumenten momentáneamente la QoS localizada sin tener que actuar sobre la propiedad QoS global (que manifiesta una gran inercia). De confirmarse la validez de estas suposiciones sería posible segmentar el espacio global en N subdominios $\{Z_i\}, 1 \leq i \leq N$ de características estadísticas similares y aplicar prioridades a las operaciones de carga automatizada y carga predictiva. Para cada dominio Z_i se pueden extraer características como se indica en (3.8) y modelar cualquier indicador de calidad global del sistema según (3.11):

$$\overline{QoS} = \sum_{i=0}^{N} QoS\left(Z_i\right) P_{req}\left(Z_i\right) \tag{3.11}$$

Con el objeto de poder experimentar con estas propiedades de la caché se ha utilizado un prototipo funcional que permite la inserción de distintos puntos de control y medición, como se describe en el Apéndice A. Se utilizarán sus posibilidades para:

- Recopilar estadísticas en tiempo real del comportamiento espacio temporal de la caché para servir de fuente de datos para los algoritmos de análisis y gestión.

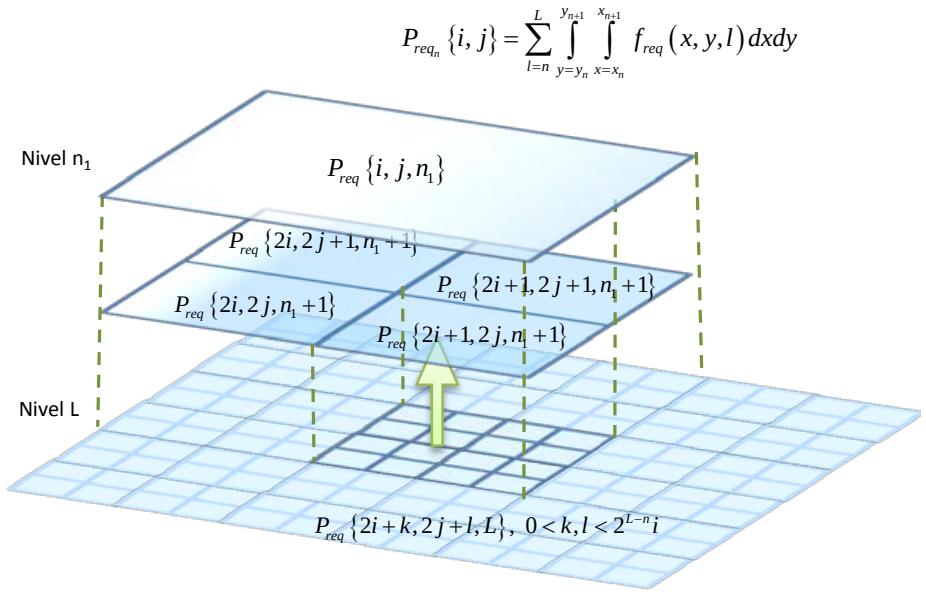


Figura 3.4: Método para extrapolar características probabilísticas usando el principio de localidad.

- Experimentar con algoritmos heurísticos y predictivos de gestión de la caché para entornos con recursos escasos o con conjunto de datos dinámicos. El objetivo indirecto es conseguir un mecanismo de autogestión del servicio de *cacheado* de teselas.
- Experimentar con técnicas de manejo transparente de las peticiones estándar WMS en entornos con servidores en cascada. En esta línea se engloban la adhesión a las interfaces estándar disponibles incluyendo la generación de metadatos secundarios y la composición de respuestas a la petición *GetFeatureInfo* a partir de las fuentes de datos primarias.
- Experimentar sobre las posibilidades, ventajas y condiciones de aplicabilidad del procesado al vuelo de teselas. Algunas operaciones podrían ser realizadas con los datos secundarios (teselas almacenadas) mediante transformaciones afines (p.ej. cambios de sistemas de referencia) o composición de teselas (p.ej. superposición de capas).

A partir de ahora, vamos a caracterizar un escenario con las siguientes condiciones de contorno:

- No es posible conseguir una caché completa, por lo que $P_h\{T(i,j,n)\} \ll 1$.
- La capacidad de procesamiento es limitada, por lo que la latencia en caso de fallo de caché es significativa. La potencial ganancia $G_c = \frac{\tau_m}{\tau_h} \gg 1$ dado que $\tau_m \gg \tau_h$.

Estas hipótesis de partida no son descabelladas puesto que incluso para implantaciones de escala modesta se generan grandes requisitos de almacenamiento. Además existe un reducido número de potenciales proveedores de información geográfica que puedan costearse un alojamiento de servidores de la escala adecuada. Por ello, estas hipótesis se utilizan como condiciones de contorno para los algoritmos propuestos en capítulos posteriores.

Capítulo 4

Estado de arte

Resumen: En este capítulo se recogen los principales estudios encontrados en la literatura relacionados con la gestión de servicios de mapas a través de la Web: transmisión de los datos, almacenamiento, indexación y políticas de gestión de caché. Muchos de estos trabajos aplican estrategias extraídas de otros ámbitos no-geográficos, como los proxies Web tradicionales, los cuáles no tienen en cuenta la componente espacial de los objetos de mapa que gestionan.

Durante los últimos años, el sector de la Información Geográfica ha experimentado un importante cambio de paradigma, evolucionando desde los Sistemas de Información Geográfica (SIG) tradicionales hacia lo que ahora conocemos como Infraestructuras de Datos Espaciales (IDEs). Este cambio persigue facilitar la disponibilidad y el acceso a datos espaciales a través de Internet. Sin embargo, esta migración de los SIG de escritorio a la Web (Web GIS) plantea nuevos e importantes desafíos.

4.1. Organización y traspaso de los datos

Los Sistemas de Información Geográfica generalmente trabajan con grandes volúmenes de datos. Por tanto, un aspecto clave es realizar el traspaso de estos datos desde los servidores remotos a los clientes en el menor tiempo posible (Peng y Tsou, 2003).

Existen múltiples alternativas para la representación de la información geográfica. Según la aproximación basada en capas, los datos espaciales se representan como un conjunto de datos temáticos (capas). Así, un mapa urbano puede estar compuesto por múltiples capas temáticas, como calles, carreteras, códigos postales, edificios, etc. Esta aproximación, ilustrada en la Figura 4.1, presenta las siguientes ventajas: (1) Procesamiento más sencillo de las consultas para el análisis espacial, y (2) Gestión más eficiente en la transmisión de los datos, pues se pueden seleccionar exclusivamente las capas de interés (Wei et al., 1999).

Wei et al. (1999) y Tu et al. (2001) propusieron también la división del mapa en teselas, dado que el volumen de los datos puede hacer impráctica la transmisión una capa en su totalidad. De esta forma, cuando un usuario solicita una determinada región del mapa, el servidor sólo transmite los datos espaciales correspondientes a las teselas que intersectan con la región solicitada. Los métodos propuestos difieren, sin embargo, en cómo definen los esquemas de teselado: Wei et al. (1999) presentan un algoritmo de teselado adaptativo que crea teselas de tamaño variable, con teselas de gran tamaño para cubrir zonas vacías y

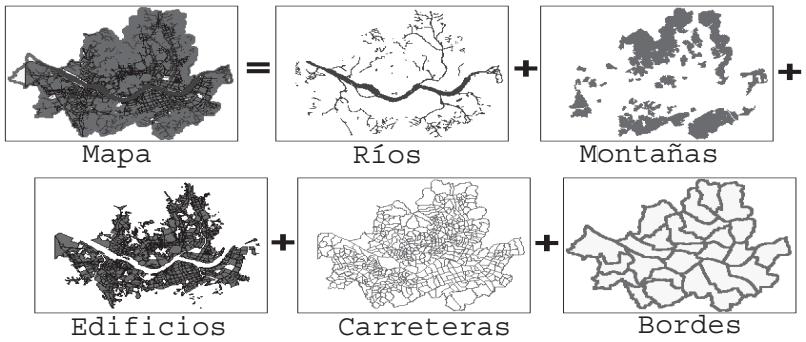


Figura 4.1: Descomposición de un mapa en capas temáticas. Imagen adaptada de (Wei et al., 1999).

teselas más pequeñas en áreas con mayor densidad de *features*, de forma equivalente a un *quadtree*. Por el contrario, Tu et al. (2001) proponen el uso de una pirámide con teselas de tamaño uniforme, realizando subdivisiones hasta que el tamaño de cada tesela en disco sea inferior a un valor dado.

El teselado adaptativo presenta claras ventajas, ya que puede establecerse el volumen de datos en cada tesela al valor óptimo para la realización de búsquedas, recuperación de datos, y otros tipos de procesamiento. Por su parte, el teselado uniforme es generalmente sub-óptimo, pero resulta ventajoso en caso de que el volumen de datos varíe con el tiempo o que éste no sea conocido de antemano, pues el coste de re-estructurar un teselado adaptativo cuando hay nuevos datos para mantener un estado óptimo puede ser sustancial (Goodchild, 1989).

Un aspecto que no contemplan los trabajos anteriores es que en los visores de mapas no todos los datos se muestran a todas las escalas de representación. Por ejemplo, al visualizar un mapa en el que se muestra todo el territorio nacional, tendría sentido mostrar los límites administrativos, pero no lo tendría el incluir el nivel de detalle de calles. Lee et al. (2003) proponen un método, que denominan como *Scale-Dependent Transmission* (Transmisión Dependiente de la Escala), cuya idea principal es "transmitir lo que se ve". Asimismo, proponen un algoritmo de generalización del mapa basado en la transformada Wavelet que adapta los datos para ajustarse a las necesidades de los diversos clientes.

Los trabajos anteriores están enfocados a la transmisión de datos vectoriales. Sin embargo, en la mayor parte de sistemas teselados que se utilizan hoy en día se transmiten imágenes ráster o *features* vectoriales rasterizadas. Estas teselas se generan a varias escalas de representación, dando lugar a pirámides de teselas (Phil Yang et al., 2005).

La distribución de teselas de mapa rasterizadas supone una mejora sustancial respecto a otras estrategias previas en las que se descargaban imágenes de gran tamaño que no serían usadas en su totalidad (Plewe, 1997). Además, la distribución de contenido pregenerado mejora notablemente la escalabilidad en comparación con servicios de mapas tradicionales, como el servicio WMS de OGC (ver Sección 2.1), en los que se genera dinámicamente y al vuelo la imagen de mapa correspondiente a la zona geográfica solicitada por los clientes.

Las teselas de mapa pueden ser almacenadas como campos BLOB (*Binary Large OBject*) en una base de datos (Barclay et al., 2000), o como archivos individuales en un sistema de ficheros. Por ejemplo, en (Talagala et al., 2000) se propone un sistema teselado para la distribución de colecciones de imágenes de gran tamaño del *Fine Arts Museum* de San

Francisco. El sistema propuesto utiliza un teselado uniforme con imágenes codificadas en formato JPEG y almacenadas en disco utilizando un *cluster* de 20 PCs y 368 unidades de disco duro.

4.2. Almacenamiento de las teselas

Un aspecto fundamental a tratar durante el diseño de un servicio de mapas teselado es el conseguir un almacenamiento óptimo de las teselas de mapa que posibilite almacenar y recuperar las imágenes de forma eficiente. Así, las teselas pueden almacenarse directamente en memoria, en el sistema de ficheros, bases de datos relacionales, la nube, o incluso en configuraciones P2P (*Peer to Peer*) distribuidas.

4.2.1. Almacenamiento en sistema de ficheros

Los sistemas de caché de los servicios web de mapas almacenan con frecuencia las imágenes de mapa directamente en el sistema de ficheros. Generalmente, cada tesela de mapa se almacena como un fichero individual del sistema operativo, y se utiliza la ruta del fichero para devolver la imagen.

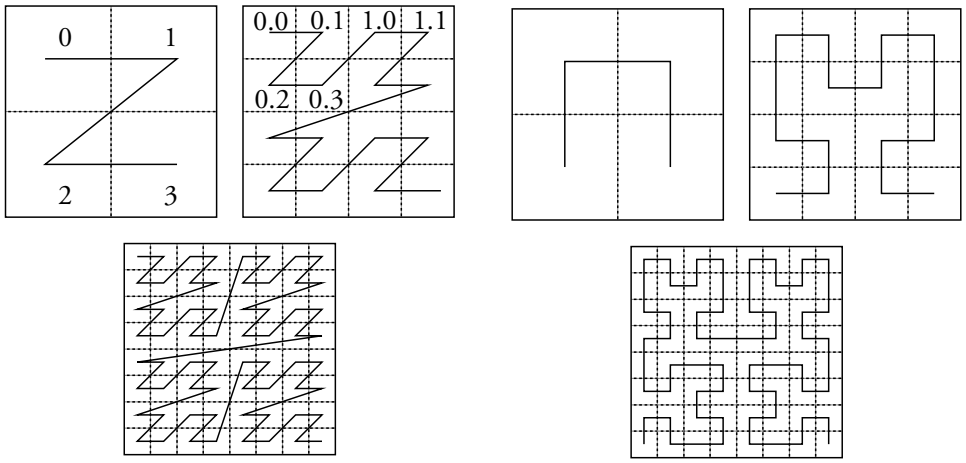
Puede aprovecharse el principio de localidad espacial para optimizar el acceso a las teselas mediante técnicas de indexado en las que la posición relativa de las mismas en la base de datos esté lo más relacionada posible con sus posiciones relativas en el espacio (Tomlinson et al., 1979).

La secuencia de Morton, también conocida como "curva Z" (Morton, 1966) es una función que permite mapear datos multi-dimensionales a una cadena uni-dimensional preservando la localidad de los mismos (ver Figura 4.2a). La secuencia de Morton para un *array* cuadrado de $n \times n$ teselas puede generarse según el siguiente algoritmo: Se numeran las filas/columnas de 0 a $n-1$, y se expresa cada número en base 2. La posición de cada tesela en la secuencia de Morton se obtiene intercalando los bits de la fila y de la columna, dando lugar a un número binario entre 0 y $n^2 - 1$.

Una alternativa a la secuencia de Morton es la curva de Hilbert. A diferencia de la anterior, la curva de Hilbert está formada por segmentos de longitud uniforme (ver Figura 4.2b). Aunque preserva mejor el orden, los cálculos son mucho más complejos. Existen múltiples estudios en la literatura (Park y Kim, 2001; Fang et al., 2008; Lee et al., 2002; Yesilmurat y Isler, 2012; Hu et al., 2011) que hacen uso de la curva de Hilbert para la precarga de teselas de mapa.

Zhang et al. (2011) proponen un sistema eficiente para el almacenamiento y recuperación de teselas de mapa en un sistema de caché. Esta eficiencia se consigue organizando las teselas de una capa en el mismo fichero, con las relaciones con las correspondencias en distintos niveles de resolución, combinada con el uso de una estructura de datos en memoria e índices en disco para facilitar la búsqueda de punteros a las teselas.

El reciente candidato a estándar OGC, GeoPackage, proporciona un contenedor abierto, no-propietario e independiente de la plataforma utilizada para la distribución y el uso directo de todo tipo de información geoespacial. Este contenedor y las API incrementarán la plataforma de interoperabilidad de aplicaciones geoespaciales y servicios web en el mundo de los móviles (Daisey, 2012).



(a) Curva Z (b) Curva de Hilbert

Figura 4.2: Curvas de relleno del espacio. Fuente: (Rigaux et al., 2002).

4.2.2. Almacenamiento en bases de datos relacionales

Las bases de datos geográficas aportan una mayor riqueza a los datos almacenados, respecto a las bases de datos convencionales, al añadir la posibilidad de asignarles una posición espacial. En (Martinez-Llario y Gonzalez-Alcaide, 2011) se recoge un listado con las extensiones espaciales más consolidadas, entre las que destaca la PostGIS (PostgreSQL), Oracle Spatial (Oracle), SpatialLite (SQLite) y MySQL Spatial (MySQL). Estas implementaciones incluyen índices espaciales que permiten realizar consultas geoespaciales con gran eficacia.

4.2.3. Almacenamiento en la nube

La computación en la nube (*Cloud Computing*) ha adquirido una gran popularidad durante los últimos años. Este paradigma posibilita el acceso ubicuo a través de la red a un conjunto configurable compartido de recursos de computación (p.e., redes, servidores, almacenamiento, aplicaciones, y servicios) que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios.

Dada la voluminosidad de los datos que gestionan y las complejas tareas computacionales que a menudo llevan a cabo, los servicios GIS son buenos candidatos para su incorporación en la nube. Así lo demuestran los numerosos estudios encontrados en la literatura; en (Xiaoqiang y Yuejin, 2010) se exponen algunas de las tecnologías principales de la computación en la nube, como DFS (*Distributed File System*), Map-Reduce y BigTable, y se propone una arquitectura *cloud* para servicios GIS, con capas de aplicación, visualización, servicios, computación, almacenamiento e infraestructura. En (Blower, 2010) se propone la implementación de un servicio WMS para cartografía raster utilizando la herramienta *Google App Engine*. En (Hefley et al., 2012) se presentan los resultados de los experimentos hechos con *Google App Engine* para aplicaciones geoespaciales en tiempo real.

En (Wang et al., 2009) se propone un *framework* para la devolución, indexado, acceso y gestión de datos espaciales en la nube. Proporcionan un modelo de objetos para datos espaciales basado en formatos de OGC WKB (*Well Known Binary*) y WKT (*Well Known*

Text), y realizan una adaptación de los algoritmos clásicos de indexado espacial, como Quad-Tree y R-Tree, para el entorno *cloud*. Implementan un prototipo del modelo propuesto sobre *Google App Engine*.

(Liu et al., 2009) proponen una aproximación para optimizar el rendimiento de las operaciones de entrada/salida de ficheros de reducido tamaño en el sistema de ficheros HDFS (*Hadoop Distributed File System*) de Hadoop. HDFS fue diseñado para gestionar ficheros de gran tamaño, produciéndose una penalización en el rendimiento al gestionar grandes cantidades de ficheros de reducido tamaño. Como consecuencia, múltiples aplicaciones GIS Web no pueden beneficiarse de este *framework*. En el trabajo anterior se propone combinar varios ficheros pequeños dando lugar a uno más grande para así reducir el número de los mismos y crear índices para cada uno de ellos. En (Bunzel et al., 2010) se propone el uso de herramientas de código abierto para ofrecer servicios GIS en la nube a bajo coste.

4.2.4. Almacenamiento distribuido P2P

Existen múltiples estudios que proponen el uso de una caché distribuida para servir los objetos en un entorno GIS Web. En (Guan et al., 2004) se exploran diversas técnicas para establecer servicios Web de GIS en entornos P2P y se propone un *framework* para estos servicios. Rui y Gui-Li (2009) proponen *SOP4GIS*, una aplicación que combina SOA (*Service-Oriented Architecture*), P2P mediante el *framework* JXTA de Sun, y PPGIS (*Public Participatory Geographic Information System*).

En (Bergamini y Haungs, 2006, 2007) se propone "*GeoTorrent*", un *proxy cache* distribuido y compartido basado en tecnología P2P. Este sistema puede funcionar en distintos modos: desde una configuración P2P totalmente distribuida, a otra completamente centralizada con un único servidor, pasando por varias configuraciones intermedias. Un cliente WMS utiliza, de forma transparente, un único nodo *GeoTorrent* a modo de servidor WMS. Como resultado, *GeoTorrent* puede controlar cómo se cachean los datos para ser compartidos entre el grupo de usuarios, cómo se precargan en base a la actividad del grupo, y cómo se ofrecen a través del servicio. Cada nodo del grupo actúa como *proxy* para uno o más clientes, y como caché para todo el grupo.

4.3. Compresión de las imágenes de mapa

Las imágenes vectoriales de mapa pueden contener ciertos elementos, como texto y objetos gráficos, que requieren un nivel de detalle muy fino. No poseen, sin embargo, tantos tonos de color como las imágenes fotográficas. Estas propiedades hacen que los métodos de compresión de imagen más adecuados para teselas vectoriales sean los métodos «*sin pérdida*» (*losseless*), tales como GIF y PNG (Fränti et al., 2002). Otra posibilidad es dividir los mapas en distintas capas de color y aplicar estándares de compresión de imagen binarios como los del Grupo 4 de ITU-T, o el actual estándar JBIG2 (Fränti et al., 2004).

Existen algunos servidores de mapas, como *RIMapperWMS* (Köbben, 2007), que permiten distribuir cartografía vectorial en formato SVG. Esto resulta especialmente adecuado para dispositivos móviles, como PDAs y *smartphones*. Por otra parte, *RIMapperWMS* incluye las llamadas *Vendor Specific Capabilities* de OGC, que en este caso posibilitan generar una salida SVG con una interfaz gráfica embebida, lo cual permite visualizar la cartografía en los clientes sin que estos necesiten disponer de un cliente WMS.

Por el contrario, para la compresión de imágenes fotográficas de mapa, como ortofotos

aéreas, los métodos de compresión «*con pérdida*», tales como JPEG, resultan más adecuados. En (Gerlek y Fleagle, 2007) se describen los retos y principales aproximaciones para la adopción del estándar JPEG2000 para distribuir imágenes geoespaciales.

4.4. Políticas de gestión de la caché

A diferencia del cacheo bajo demanda, los mecanismos de precarga o *seeding* generan y almacenan los objetos con anterioridad a que éstos sean solicitados por los usuarios. Esto se hace anticipando que es probable que estos objetos sean pedidos en el futuro, de forma que dichos accesos pueden ser satisfechos muy rápidamente a partir de la caché en lugar de devolverlos desde el servicio WMS original. Los algoritmos de precarga pueden clasificarse en función de si trabajan a corto o a largo plazo (Venkataramani et al., 2002). En la precarga a largo plazo, o *long-term prefetching*, se utilizan los patrones globales de acceso de los objetos para identificar una colección de objetos "valiosos" que deberían introducirse en la caché. En la precarga a corto plazo, o *short-term prefetching*, la caché de mapa utiliza el histórico reciente con los accesos de clientes individuales para predecir y precargar aquellos objetos que se espera sean pedidos a corto plazo.

A continuación se recogen los principales trabajos encontrados en la literatura relacionados con la gestión de cachés de mapas: precarga inicial, carga dinámica y estrategias de reemplazo.

4.4.1. Políticas de precarga inicial (*long-term*)

En Quinn y Gahegan (2010) se propone un modelo predictivo para determinar zonas prioritarias de cacheo. Este modelo considera como entradas estos cuatro conjuntos de datos vectoriales: lugares poblados, carreteras, zonas costeras y puntos de interés. Estos conjuntos corresponden con los tipos de lugares que se ha determinado visualmente son más visitados en la aplicación Web Hotmap (Fisher, 2007a,b, 2009).

En la Figura 4.3 se muestra el *workflow* utilizado por este modelo, implementado sobre ArcGIS:

1. Se eliminan los polígonos y agujeros de pequeño tamaño. Durante esta etapa se toman decisiones acerca de (por ejemplo) si se rellenan los pequeños agujeros situados en regiones amplias seleccionadas para el cacheo, dada la elevada probabilidad de que los usuarios naveguen a través de dicho área, o si se mantienen o no pequeñas poblaciones en regiones no incluidas como prioritarias.

2. La salida se combina como una única *feature* de gran tamaño, en vez de múltiples polígonos independientes. Esto permite un funcionamiento más eficiente del *software* de cacheo.

3. Se recorta la salida a la zona de interés.

4. El modelo simplifica la geometría de los bordes del polígono utilizando un algoritmo para la eliminación de puntos sobrantes. La salida del modelo debe intersectar con la malla de teselado del perfil utilizado, para determinar qué teselas deberían ser generadas. Dado que la resolución de esta malla es más gruesa que la salida del modelo, no es necesario que ésta contenga un contorno con mucho detalle. Además

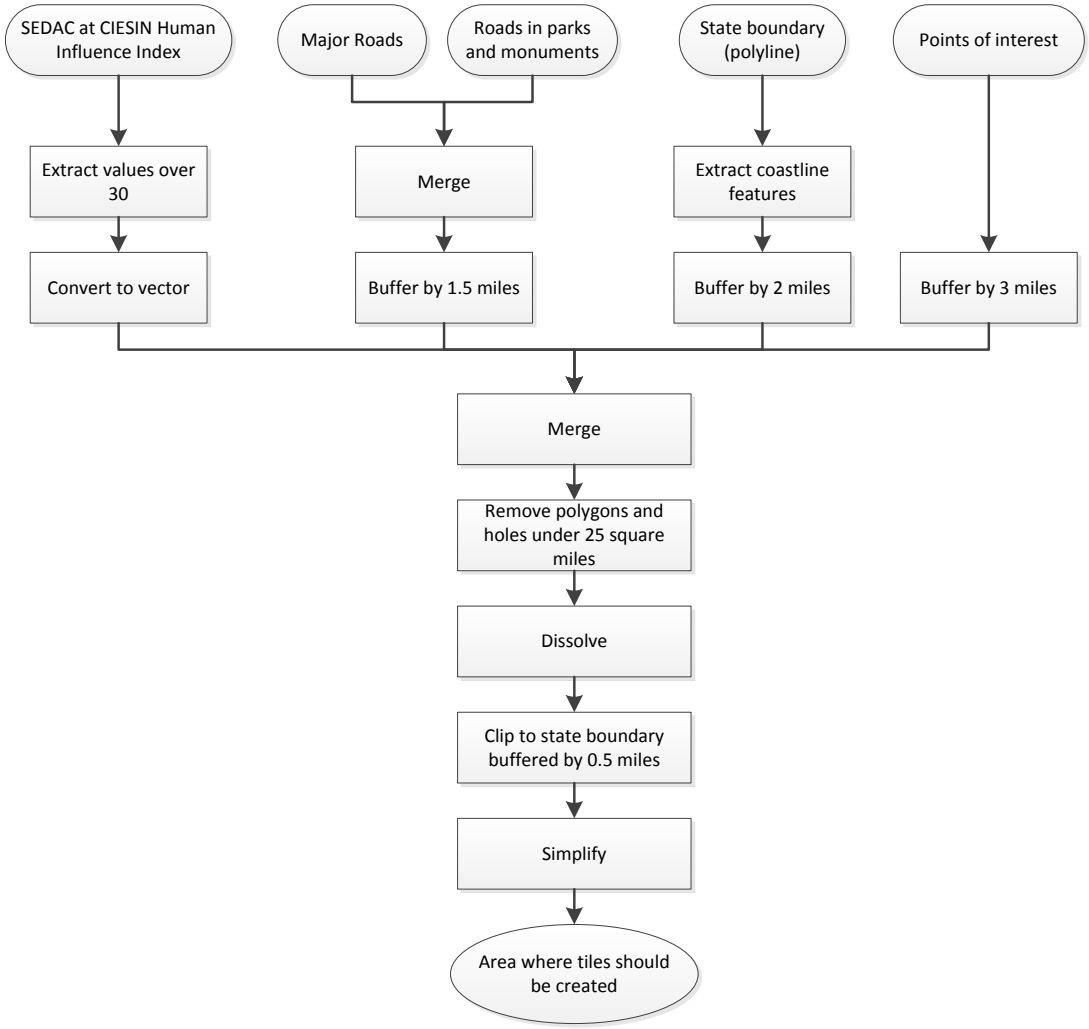


Figura 4.3: *Workflow* para determinar las áreas prioritarias para el cacheo. Imagen adaptada de (Quinn y Gahegan, 2010).

el rendimiento del *software* utilizado para el cacheo se degrada cuando se utiliza una *feature* con un nivel de detalle elevado.

La salida del modelo es un vector que muestra qué teselas deberían ser pregeneradas. No se realiza una ponderación de los cuatro parámetros de entrada, lo que significa que si un determinado lugar satisface cualquiera de los criterios se incluye en la salida final. Dependiendo de la cantidad de recursos disponibles para la creación y almacenamiento de las teselas, se pueden ajustar ciertos parámetros, como las distancias de *buffer* o algunos umbrales, para crear más o menos teselas.

4.4.2. Políticas de carga dinámica (*short term*)

En Lee et al. (2002) usan una NSMC (*Neighbor Selection Markov Chain*) para predecir los posibles movimientos subsiguientes de un determinado usuario en base a los k-movimientos anteriores. Al igual que en el trabajo anterior, en (Li et al., 2012) también se

emplean cadenas de Markov para estimar las probabilidades de transición de las teselas. Este trabajo considera tanto las características de los accesos a corto como a largo plazo y utiliza un sistema de caché basado en *clusters*. Yesilmurat y Isler (2012) emplean una aproximación heurística para devolver las posibles teselas siguientes a una pedida de acuerdo a la navegación previa del usuario, asumiendo un movimiento inercial y asignando mayor importancia a los últimos movimientos.

Kefaloukos et al. (2012) proponen un *framework* para la precarga de teselas llamado *TileHeat*. Este *framework* emplea un algoritmo predictivo que utiliza los *heatmaps* de los instantes de tiempo $t - n$ a t para predecir el *heatmap* en el instante $t + 1$. Las teselas del *heatmap* estimado se ordenan por orden descendente de popularidad, y las k teselas con mayor probabilidad de ser pedidas que no estén en la caché son seleccionadas para ser precargadas. La estimación del *heatmap* futuro en base a los anteriores se realiza mediante el método de suavizado exponencial *Holt-Winters*. Sin embargo, el método anterior tan sólo tiene en cuenta las teselas que aparecen en los registros de entrenamiento. Por ello, proponen una etapa previa de disipación espacial, aprovechando la localidad espacial de los objetos enunciada por Tobler (1970). En (Park y Kim, 2001) se propone un mecanismo para la precarga de objetos utilizando el principio de localidad espacial y la ordenación proporcionada por la curva de Hilbert.

En (Kang et al., 2001) se proponen sendos algoritmos de precarga y de limpieza de teselas destinados a servicios GIS proporcionados a través de la Web, en los que el servidor maneja y sirve distintas versiones de los mismos datos de mapa en distintos niveles de abstracción para múltiples escalas. El algoritmo de precarga propuesto predice qué teselas serán solicitadas basándose en el patrón global de accesos de los usuarios a largo plazo e introduce estas teselas en la caché. Para ello, se recogen las probabilidades de transición entre teselas adyacentes. Con estas probabilidades puede predecirse qué teselas tienen mayor probabilidad de ser pedidas a continuación, y precargando estas teselas recomendadas pueden devolverse las peticiones de los usuarios a partir de la caché más rápidamente.

Plantean una arquitectura genérica para servicios GIS cuyos componentes se reproducen en la Figura 4.4. Los principales componentes de esta arquitectura se distribuyen en dos grandes grupos: cliente y servidor. Un cliente es un explorador Web con varias capacidades de procesado. El servidor gestiona una base de datos GIS que puede ser tanto espacial como no-espacial, manejando la información espacial en forma de teselas. Un mapa se descompone en un conjunto de teselas que pueden ser transferidas a los clientes en un tiempo reducido.

Las peticiones de los usuarios se clasifican como comandos de exploración y consultas. Estas últimas permiten devolver la tesela que contiene el punto cuyas coordenadas se especifican, el grupo de ellas contenidas en una determinada región, o las que cumplen unos determinados requisitos. Los comandos de exploración incluyen los desplazamientos en el plano y las operaciones de aumento/disminución del nivel de *zoom*. Generalmente, las consultas constituyen la primera petición del usuario al servicio, y en base a su respuesta se envía una secuencia de comandos de exploración y/o consultas. Mediante las operaciones de desplazamiento en el plano los usuarios pueden moverse hacia una de las cuatro teselas adyacentes.

Cuando un usuario envía una petición al cliente, el componente QAE (*Query Analyzer and Executor*) analiza la petición y solicita al gestor de caché (CM - *Cache Manager*) los datos necesarios. Si el gestor de caché puede encontrar los datos en caché se envían directamente a partir de la misma y en caso contrario se redirige la petición al Agente de Precarga (PA - *Prefetch Agent*), que a su vez redirige la petición al componente PE (*Prefetch*

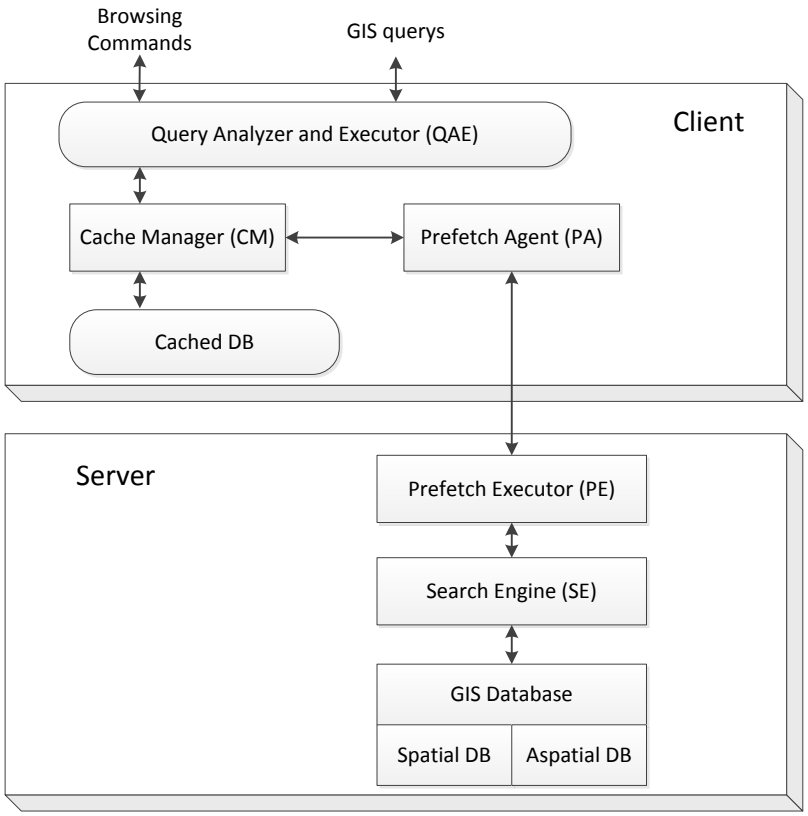


Figura 4.4: Arquitectura de un servicio Web GIS. Imagen adaptada de (Kang et al., 2002).

Executor) del servidor añadiendo información adicional. Esta información adicional incluye la lista de teselas que ya han sido cacheadas en el cliente, las frecuencias de transición entre las teselas introducidas en caché desde la última consulta al servidor y el número de teselas que se desea precargar. PE descompone la petición en la petición original recibida por el PA (*Prefetch Agent*) y la información adicional añadida por este último. La petición original se envía al motor de búsqueda (SE - *Search Engine*) para devolver el resultado de la petición del usuario. A continuación se realiza la precarga en base al objeto devuelto y a la información adicional. Todo ello es devuelto por PE hacia el CM a través de el PA. Finalmente, cuando el QAE recibe el resultado devuelto, ejecuta la operación del usuario sobre los datos devueltos y envía al usuario el resultado final.

Para determinar las teselas que deben ser precargadas, el PE actualiza el patrón global de acceso mediante el patrón local enviado por el PA. Si el número de teselas devueltas como resultado de la petición es mayor que el número de teselas que se desea precargar, no se realiza precarga. En caso contrario, se calculan las probabilidades normalizadas de transición de la tesela solicitada a sus cuatro vecinas. Si el tamaño de precarga es mayor que 1, se calculan las probabilidades de transición hacia aquellas teselas situadas a una distancia no-superior a este tamaño. Posteriormente se ordenan estas probabilidades de forma descendente y se trunca el resultado al número máximo especificado por el PA. Finalmente, se eliminan del resultado las teselas que ya han sido cacheadas en el cliente debido a peticiones previas. Asimismo, se devuelve la lista con las probabilidades de transición de las teselas, para que el gestor de caché la utilice en el algoritmo de reemplazo.

En (Kang et al., 2002), los mismos autores extienden el algoritmo propuesto en (Kang et al., 2001) para tener en cuenta también las operaciones de aumento y disminución de zoom.

En (Zhang, 2008) se presenta una aproximación para la mejora de rendimiento en la representación de marcadores en *mashups* de mapas. Esta aproximación se basa en un modelo de precarga mediante la predicción de las regiones de mapa a las que posiblemente se llegará en movimientos sucesivos durante la navegación del usuario. Los movimientos de los usuarios se clasifican según diversos patrones. El modelo comprueba el histórico de peticiones realizadas por un usuario específico, encuentra posibles patrones que puede estar describiendo, y entonces predice los posibles movimientos sucesivos del usuario de acuerdo al algoritmo específico que responde a dicho patrón.

La utilización de este modelo implica asumir las siguientes suposiciones:

1. Generalmente, los usuarios no cambian constantemente sus patrones de movimiento, incluyendo la dirección y la distancia.

2. Cuanto más rápido se mueven los usuarios sobre el mapa, es más improbable que cambien su patrón de movimiento.

3. En la mayor parte de los casos, la distancia de un movimiento no excede aquella cubierta por una pantalla de mapa.

En este estudio se contemplan dos aproximaciones para implementar esta herramienta de precarga. Ambas necesitan capturar las peticiones HTTP de los navegadores, analizar estas peticiones y realizar la precarga basándose en el análisis con el modelo propuesto. Una posibilidad propuesta es la de embeber la herramienta en el explorador Firefox utilizando código Javascript mediante la extensión GreaseMonkey, y usar el *script* para captura la petición XMLHTTPRequest, analizarla y realizar la precarga si es necesario. El inconveniente de esta aproximación es que la información precargada se almacena en la memoria destinada para el navegador, consumiendo sus recursos y ralentizando su funcionamiento. Además, esta aproximación solo es aplicable para Firefox, pues otros exploradores no disponen de una extensión equivalente. Por ello, una aproximación más interesante es la utilización de un servidor independiente, en el lado del cliente, que funcione como un *Proxy* capturando las peticiones HTTP y realizando la precarga.

A pesar de que este estudio se particulariza para la aplicación concreta de la representación de marcadores en *mashups* de mapa, debe advertirse que su aplicación es inmediata en la aceleración de servicios de mapas teselados, objetivo de esta investigación. Sin embargo las diferencias que guarda con éste son muy significativas, consecuencia de que, mientras que la solución propuesta en (Zhang, 2008) está del lado del cliente, aquí se propone una solución situada en el lado del servidor. Mediante una aproximación del lado del cliente se busca maximizar la QoS de un sólo usuario. Por el contrario, las soluciones del lado del servidor buscan maximizar la QoS global de todos los usuarios, dada la dificultad de identificar los usuarios individuales que originaron las peticiones.

En (Kirchner et al., 2004; Cho, 2002; Fang et al., 2008) se propone que puede aprovecharse el conocimiento de la localización de los usuarios para anticipar el contenido que éstos van a solicitar y anticiparse realizando precarga de las teselas que se espera se van a realizar en un futuro cercano. Trasladando esta estrategia a nuestro contexto, se extrae que podría aprovecharse la posición de la vista actual del mapa para predecir futuros desplazamientos sobre el mismo.

4.4.3. Políticas de reemplazo

La mayor parte de las estrategias de reemplazo en cachés Web encontradas en la literatura se basan en el uso de alguna de estas características de los objetos que gestionan: *actualidad* (tiempo desde que se produjo la última consulta al objeto), *frecuencia* (número de peticiones del objeto), *tamaño* del objeto Web y *coste* para obtener el objeto a partir de su servicio de origen. En función de cuáles de estas características se utilicen, las estrategias de reemplazo pueden clasificarse como estrategias temporales, de frecuencia, una combinación de ambas, las que utilizan funciones arbitrarias, y las estrategias aleatorias (Podlipnig y Böszörmenyi, 2003).

Las primeras explotan la localidad temporal de referencia observada en las peticiones Web. Se trata generalmente de extensiones de la popular estrategia LRU (*Least Recently Used*), la cuál elimina de la caché el objeto menos referenciado recientemente. El algoritmo PSS (*Pyramidal Selection Scheme*) (Aggarwal et al., 1999) utiliza una clasificación piramidal de los objetos dependiendo de su tamaño, de forma que se mantiene una lista LRU independiente para cada uno de los grupos. Tan solo se comparan los objetos menos referenciados recientemente de cada grupo, siendo seleccionado para el reemplazo el objeto que maximice el producto de su tamaño y el número de accesos a la caché desde la última vez que se recibió una petición de dicho objeto.

Las estrategias basadas en frecuencia se apoyan en el hecho de que la popularidad de los objetos está relacionada con la frecuencia con que son pedidos. Estas estrategias se construyen en torno a la estrategia LFU (*Least Frequently Used*), la cuál elimina de la *caché* el objeto que ha sido pedido con menor frecuencia.

Existen estrategias que combinan la información temporal y frecuencial para tomar decisiones de reemplazo.

Otras estrategias utilizan una función general de varios parámetros para la toma de decisiones sobre qué objetos borrar de la caché. Por ejemplo, la estrategia GD-Size (Cao y Irani, 1997) utiliza una función del tamaño, coste y un factor temporal. El algoritmo GDSF (Arlitt et al., 2000) es una extensión del anterior que tiene también en consideración el parámetro frecuencial. La estrategia *Least-Unified Value* (LUV) (Bahn et al., 2002) califica un objeto en función de sus referencias pasadas para estimar la probabilidad de recibir una petición en el futuro, y normaliza este valor con el coste del objeto por unidad de tamaño.

Por último, las estrategias aleatorias utilizan una aproximación no-determinística para seleccionar de forma aleatoria un objeto candidato para el reemplazo.

En (Podlipnig y Böszörmenyi, 2003) se presenta un estudio exhaustivo de las estrategias de reemplazo en las cachés Web. Según este trabajo, algunos algoritmos como GD-Size, GDSF, LUV y PSS pueden considerarse como "suficientemente buenos" para las necesidades requeridas cuando este trabajo fue publicado en 2003. Sin embargo, la explosión de tráfico web de mapas no se produjo hasta unos pocos años después.

A pesar de la enorme proliferación de algoritmos de reemplazo para las cachés Web, existe un reducido número de políticas de reemplazo específicas para las *cachés* de teselas, que se beneficien de la correlación espacial entre las peticiones. Los algoritmos de reemplazo de una caché de teselas de mapas pueden aprovechar el principio de localidad espacial formulado por Tobler como la primera ley de la geografía, que dice que *"Todas las cosas están relacionadas entre sí, pero las cosas más próximas en el espacio tienen una relación mayor que las distantes."* (Tobler, 1970).

En (Kang et al., 2001) y (Kang et al., 2002) se propone un algoritmo de precarga basa-

do en estimaciones de probabilidad, así como un algoritmo de reemplazo colaborativo para sistemas web de información geográfica (*Web GIS*). El servidor recolecta y mantiene las probabilidades de transición entre teselas adyacentes. Con estas probabilidades, el servidor es capaz de predecir qué teselas tienen una mayor probabilidad de ser pedidas a continuación, basándose en los patrones de acceso globales de todos los usuarios y la semántica de las peticiones. El algoritmo de reemplazo de caché propuesto determina qué teselas deberían ser reemplazadas basándose en las estimaciones de probabilidad de los accesos futuros.

Por otra parte, en la literatura se han encontrado múltiples estudios basados en el uso de algoritmos de aprendizaje automático, como redes neuronales y algoritmos genéticos, aplicados al reemplazo de cachés Web (Venketesh y Venkatesan, 2009).

El uso de redes neuronales para la toma de decisiones de reemplazo en una caché fue introducido inicialmente por Khalid (1997a) y el algoritmo KORA. Este algoritmo utiliza una red neuronal de tipo *backpropagation* con el propósito de asistir en la toma de decisiones acerca del reemplazo de líneas o bloques en caché. KORA identifica y posteriormente descarta las líneas muertas en las memorias de caché. Se basa en un trabajo anterior de Pomerene et al. (1984), quien sugirió el uso de un directorio fantasma para contemplar un histórico mayor en la toma de decisiones mediante la estrategia LRU. Más tarde se propuso una versión mejorada del anterior conocida como KORA-2 (Khalid y Obaidat, 1999b; Khalid, 1997b). Por otra parte, existen otros algoritmos basados en KORA (Obaidat y Khalid, 1998; Khalid y Obaidat, 1999a).

ElAarag y Romano (2009b); Romano y ElAarag (2011); ElAarag y Cobb (2006); ElAarag y Romano (2009a); Cobb y ElAarag (2008) proponen también el uso de una red neuronal de retropropagación ó *backpropagation* para la toma de decisiones de reemplazo en un *proxy web caché*. En Tian et al. (2002) se presenta un predictor que aprende a reconocer los patrones de peticiones de páginas Web y es capaz de predecir accesos futuros. Por último, en El Khayari et al. (2009) se propone el uso de una red neuronal para soportar la adaptabilidad del algoritmo de *caché* C-LRU (*Class-based Least Recently Used*).

A pesar de la multitud de trabajos encontrados en la literatura en los que se aplican redes neuronales para el reemplazo de objetos en cachés Web convencionales, no se ha encontrado ningún estudio similar aplicado a una caché de mapas. En el Capítulo 6.1 de esta tesis se propone un nuevo algoritmo de reemplazo de caché mediante redes neuronales que considera además la naturaleza espacial de los datos gestionados por un servicio de mapas.

Capítulo 5

Caracterización de tráfico de servicios de mapas teselados

Using hits and page views to judge site success is like evaluating a musical performance by its volume.

Eric Schmitt (1999)

Resumen: Para poder mejorar el rendimiento de los servicios de mapas resulta fundamental conocer, en primer lugar, la carga de trabajo de estos servicios. En este capítulo se presenta un estudio de los patrones de acceso a diversos servicios de mapas teselados de ámbito público. A lo largo de este estudio, se hace especial hincapié en la búsqueda de invariantes en la carga de trabajo: observaciones que se repiten en todos los conjuntos de datos bajo estudio. A pesar de tratarse de un caso particular de servicio Web, dada la naturaleza espacial de los objetos que gestionan, los servicios de mapas presentan nuevos patrones que no aparecen en los servicios Web convencionales.

La existencia de patrones espacio-temporales entre las peticiones de los usuarios puede utilizarse para el desarrollo de mecanismos que permitan optimizar la gestión de los sistemas de caché de mapas teselados:

- Desarrollar algoritmos de actualización de caché en tiempo real mediante el análisis predictivo de las peticiones de una fuente de datos WMS-C.
- Desarrollar algoritmos adaptativos de precarga y limpieza de caché con el objeto de disminuir el tiempo de inicio de servicio y maximizar la calidad de servicio en relación al tiempo.

Resulta por tanto de especial relevancia caracterizar con el mayor detalle posible la carga de trabajo de estos servicios. Sin embargo, a pesar de que pueden encontrarse en la literatura numerosos estudios sobre la caracterización de la carga en servidores Web convencionales, no se han encontrado estudios equivalentes particularizados a servicios de mapas.

A continuación se presenta un estudio detallado de la carga de trabajo de diversos servicios de mapas teselados. Este estudio se centra en la búsqueda de invariantes en los patrones de acceso de los servicios de mapas analizados y en el descubrimiento de divergencias respecto a los patrones encontrados en los servicios Web convencionales.

5.1. Registros de Acceso (Logs)

En una configuración típica de un servicio de mapas en un entorno de producción, los usuarios no realizan las peticiones HTTP directamente al servidor de mapas. Generalmente el acceso se realiza a través de uno o varios *proxies* Web que redireccionan las peticiones al servidor de mapas, ocultando la dirección final del servicio a los clientes. Una funcionalidad básica de la que disponen la mayor parte de los servidores Web es la de mantener un registro o *log* de la información relativa a las peticiones de los usuarios.

En este estudio, los *logs* recogidos corresponden a los registros de acceso estándar de Apache con información sobre todas las peticiones que procesa, utilizando el formato CLF (Formato Común de Registro - *Common Log Format*) (Apache Software Foundation, 2011).

A continuación se muestra, a modo de ejemplo, el registro de una petición de mapas expresada en este formato:

```
193.144.251.29 - - [01/Jun/2008:02:04:11 +0200]
"GET /wms-c/CARTOCIUDAD/CARTOCIUDAD?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=Todas
&FORMAT=image/png&SRS=EPSG:4326&BBOX=-11.25,33.75,0,45&WIDTH=256&HEIGHT=256&STYLES=
&EXCEPTIONS=application/vnd.ogc.se_inimage HTTP/1.1" 200 10495
```

La información extraída de estos registros es la siguiente:

- `193.144.251.29` - Dirección IP o *hostname* del cliente (host remoto) que hizo la petición al servidor. Las direcciones IP que se registran no son necesariamente las direcciones de las máquinas de los usuarios finales. Si existe un servidor proxy entre el usuario final y el servidor, la dirección que se registra es la del proxy.
- `01/Jun/2008:02:04:11 +0200` - Fecha en la que el servidor recibió la petición, con precisión de segundos.
- `200` - Código de estado que el servidor envía de vuelta al cliente. Esta información es muy valiosa, porque revela si la petición fue respondida con éxito por el servidor (los códigos que empiezan por 2), una redirección (los códigos que empiezan por 3), un error provocado por el cliente (los códigos que empiezan por 4), o un error en el servidor (los códigos que empiezan por 5).
- `10495` - Tamaño del objeto retornado al cliente, no incluídas las cabeceras de respuesta, expresado en *bytes*.

De la línea de la petición del cliente se extraen los parámetros propios de la petición WMS-C: *service*, *version*, *request*, *layers*, *width*, *height*, *format*, *styles*, *exceptions* y *crs* (ver Sección 2.1).

En el presente trabajo se ha creado un sistema para la extracción, normalización y almacenamiento de los datos procedentes de estos *logs*, para su posterior explotación. Dado el elevado volumen de registros a procesar se ha establecido como requisito primordial que la herramienta automatice en la mayor medida posible este proceso. Para ello, el sistema ha sido creado en torno a un CMS *Open Source*: *Alfresco*[1]. *Alfresco* ofrece múltiples protocolos para subida/bajada de ficheros, como HTTP, HTTPS, CIFS/Samba, FTP y WebDav. La subida de un fichero de *logs* a la plataforma desencadena la ejecución de un *workflow* jBPM[2]

[1] http://www.alfresco.com/
[2] http://www.jboss.org/jbpm

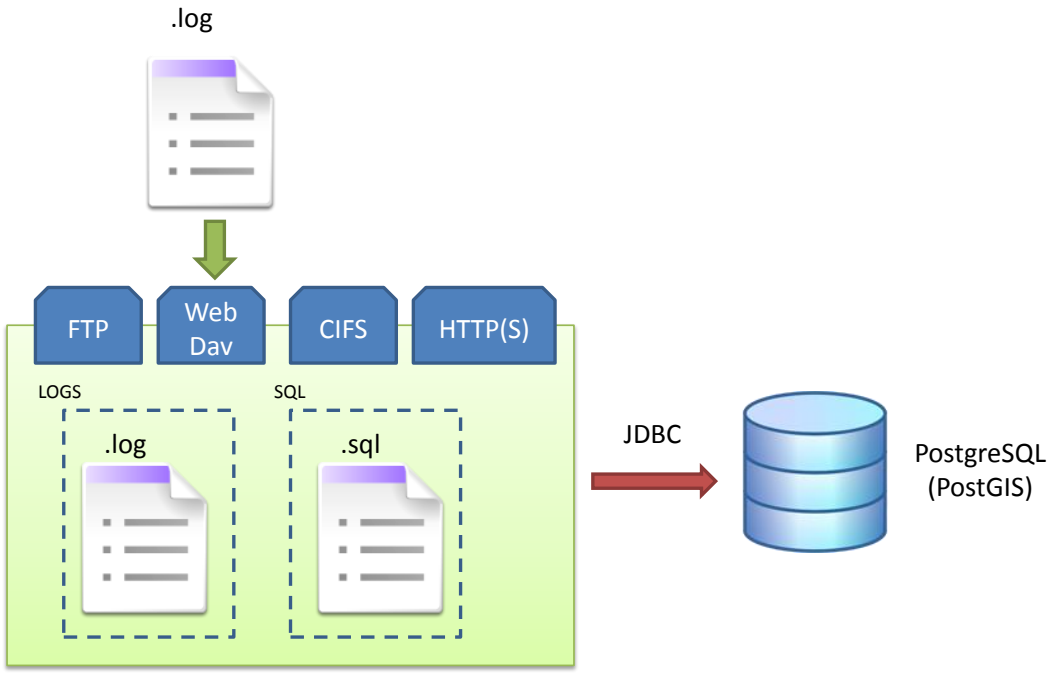


Figura 5.1: Escenario para la extracción, normalización y almacenamiento de los datos procedentes de los históricos de peticiones a los servidores de mapas.

que realiza la inserción de los datos de las peticiones en una base de datos PostgreSQL con la extensión espacial PostGIS. También se mantiene un fichero .sql en el repositorio para posibles necesidades futuras (ver Figura 5.1).

Asimismo, se ha desarrollado un programa en *Shell Script* para la extracción de los datos de las peticiones y la creación de las consultas SQL para insertar los datos en PostgreSQL/-PostGIS. El código de este programa está disponible en `http://pastebin.com/5sXfvMmg`.

5.2. Servicios de Mapas Analizados

Durante el presente estudio se han utilizado registros de peticiones realizadas a los servicios WMS-C de *Cartociudad*, *IDEE-Base* y *PNOA*, facilitados por el Instituto Geográfico Nacional (IGN).

Cartociudad[3] es el resultado de la integración y armonización de datos aportados por diferentes organismos públicos (principalmente Dirección General del Catastro, Instituto Nacional de Estadística, Sociedad Estatal de Correos y Telégrafos e Instituto Geográfico Nacional) que ha dado lugar a un sistema de información geográfica de red viaria continua, e información parcelaria, censal y postal, cuyo ámbito es todo el territorio nacional.

El servicio *IDEE-Base* permite visualizar la Base Cartográfica Numérica 1:25.000 y 1:200.000 del IGN, que contiene temas de información como hidrografía, vías de comunicación, núcleos y población, toponimia, transporte, etc.

Por último, el servicio de mapas *PNOA* permite visualizar las ortofotos del Plan Nacional

[3] http://www.cartociudad.es/portal/1024/index.htm

Tabla 5.1: Parámetros de interés de los servicios WMS-C analizados.

	CARTOCIUDAD	PNOA	IDEE_BASE
layers	DivisionTerritorial_Poligono CodigoPostal Callejero Todas SeccionCensal	PNOA	Todas
srs	EPSG:4258	EPSG:4258	EPSG:4326
format	image/png	image/jpeg	image/jpeg
width	256		
height	256		
bbox	minx=-180 miny=-90 maxx=180 maxy=90		
scales	0.70312500000000000000 0.08789062500000000000 0.01098632812500000000 0.00137329101562500000 0.00017166137695312500 0.00002145767211914063 0.00000268220901489258	0.35156250000000000000 0.04394531250000000000 0.00549316406250000000 0.00068664550781250000 0.00008583068847656250 0.00001072883605957031 0.00000134110450744629	0.17578125000000000000 0.02197265625000000000 0.00274658203125000000 0.00034332275390625000 0.00004291534423828125 0.00000536441802978516

de Ortofotografía Aérea, cuyo objetivo es obtener una cobertura ortofotográfica bienal de todo el territorio nacional.

Las versiones teseladas de estos servicios utilizan sendas cachés de teselas implementadas mediante *TileCache*, de Metacarta. Este sistema de caché sigue la especificación WMS-C de OSGeo.

La Tabla 5.1 muestra los parámetros principales de estos servicios extraídos del documento de metadatos del servicio[4], resultado de invocar una operación de tipo *getCapabilities*.

Como se puede observar, los tres servicios presentan el mismo esquema de teselado, con 20 escalas soportadas. Se trata de un subconjunto del *Well-known scale set* `GlobalCRS84Scale` definido en (Masó et al., 2010). El primer nivel soportado permite representar toda la tierra mediante dos teselas de 256 x 256 píxels (una para las longitudes negativas y otra para las positivas). Cada tesela de un nivel de resolución da lugar a cuatro teselas en el siguiente nivel, por lo que el número de éstas crece en potencias de 4.

5.3. Datos crudos

En la Tabla 5.2 se recoge un sumario con las estadísticas globales de los registros de peticiones a los servicios WMS-C considerados.

El servicio *IDEE-Base* es el que recibe un mayor número de peticiones diarias (182565), casi un orden de magnitud por encima de las recibidas por *Cartociudad* (19577).

Se aprecia un porcentaje significativo (11 %) de peticiones erróneas en el servicio de *Cartociudad*: HTTP 400 (0.16 %), 404 (8.71 %) y 500 (2.13 %).

[4]Estos metadatos son los correspondientes a los registros analizados. Estos servicios han experimentado cambios desde entonces, soportando ahora nuevas capas, otros sistemas de referencia de coordenadas, etc.

Tabla 5.2: Estadísticas globales de los registros de peticiones a los servicios WMS-C

	IDEE_BASE	CARTOCIUDAD	PNOA
total peticiones	16978535	3778369	9816747
válidas	16891385 (99.49 %)	3362940 (89 %)	9678358 (98.59 %)
erróneas	87150 (0.51 %)	415429 (11 %)	138389 (1.41 %)
GetMap	16978026 (99.997 %)	3683101 (97.479 %)	9816431 (99.997 %)
GetCapabilities	509 (0.003 %)	95260 (2.521 %)	316 (0.003 %)
GetFeatureInfo	0 (0 %)	8 ($\approx$ 0 %)	0 (0 %)
fecha inicio	2010-03-15	2009-12-09	2010-03-19
fecha fin	2010-06-17	2010-06-20	2010-06-17
duración	93 días	193 días	89 días
frecuencia	182565 petic/dia	19577 petic/dia	110300 petic/dia
tamaño objetos	6365.574 bytes	29181.239 bytes	12549.867 bytes

5.4. Reducción de los datos

Dado que únicamente las peticiones satisfactorias son responsables de la transferencia de documentos por parte del servidor, tan sólo se usan los registros correspondientes a estas peticiones durante el análisis que se llevará a cabo a continuación. Por otra parte, tan sólo estamos interesados en las peticiones de mapas (*GetMap*), por lo que se han excluido para este estudio el resto de peticiones: *GetFeatureInfo* y *GetCapabilities*.

5.5. Concentración de las peticiones

El primer estudio llevado a cabo se centra en la frecuencia de accesos para los distintos objetos. Como resulta de esperar, no todas las teselas recaban el mismo interés por parte de los usuarios, sino que mientras que existen teselas de gran popularidad que se piden con mucha frecuencia y en intervalos cortos de tiempo por muchos clientes, otras no se piden casi nunca, si es que lo hacen en alguna ocasión. Para ilustrar este comportamiento no-uniforme de accesos, se ha ordenado la lista de teselas distintas solicitadas por orden descendente en función del número de veces que son accedidas, mostrando entonces la frecuencia acumulada de las peticiones frente a la fracción de teselas referenciadas. El gráfico resultante para cada uno de los servicios analizados se muestra en la Figura 5.2.

En dicha figura se ilustra este comportamiento no-uniforme en el acceso a las teselas. En el caso del servicio IDEE-Base, un 10 % de las teselas son responsables de casi un 90 % de las peticiones recibidas. Este servicio es el que presenta una mayor concentración de las peticiones, mientras que el servicio PNOA es el que muestra una concentración menor.

Este tipo de distribuciones resulta muy beneficioso para la actuación de mecanismos de caché, ya que puede conseguirse una elevada tasa de aciertos manteniendo en caché una reducida fracción de los objetos.

En múltiples trabajos encontrados en la literatura, se ha demostrado que el patrón de acceso a las teselas satisface la conocida «regla 20/80» de la sociología (Koch, 1999). En (Fisher, 2007b) se encontró que la relación entre la probabilidad de acceso de una tesela y su posición en el *ranking* sigue una distribución de tipo Zipf (Zipf, 1949). Del mismo modo, Talagala et al. (2000) encontraron este tipo de distribución al analizar la popularidad de las teselas de un servicio para la distribución de colecciones de imágenes de obras de arte de gran tamaño.

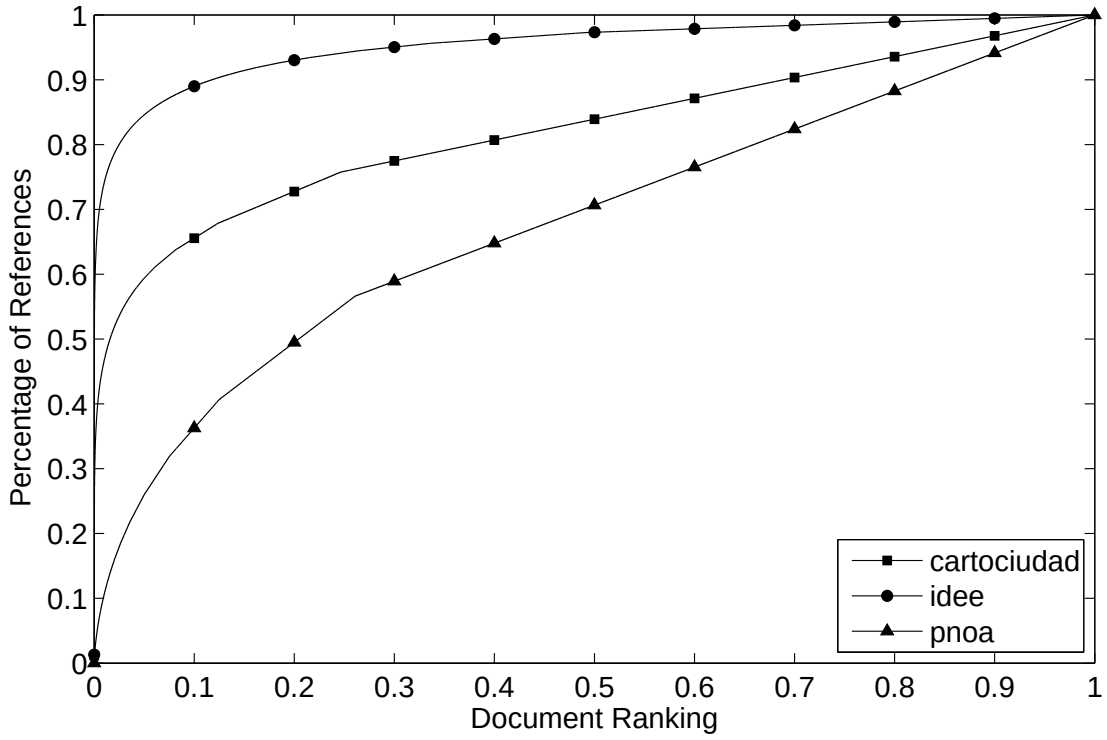


Figura 5.2: Función de distribución acumulada (CDF) de accesos a las teselas.

Antes de entrar en detalles sobre la distribución de peticiones de nuestros servicios de mapas, conviene aclarar los siguientes conceptos: «Zipf», «power-law» y «Pareto». Estos términos aparecen con frecuencia en múltiples estudios de caracterización de tráfico Web (Arlitt y Williamson, 1996; Breslau et al., 1999), y pueden generar una gran confusión. Los tres términos anteriores se utilizan para describir fenómenos en los que hay unos pocos sucesos que se repiten mucho y muchos que se repiten poco.

George Zipf describió el comportamiento estadístico de la distribución de las palabras en los textos. Propuso que en un texto cualquiera, existe una relación matemática entre la frecuencia absoluta de cada palabra $f(r)$ y la posición o *ranking* r que ocupa en el listado de las palabras usadas en el texto, por orden descendente de frecuencia. La relación que encontró puede expresarse mediante la siguiente expresión:

$$f(r) = \frac{C}{r^{\alpha}} \tag{5.1}$$

siendo f la frecuencia, r el *ranking*, C y α constantes.

Para detectar si la distribución de peticiones a nuestros servicios de mapas sigue una ley de potencias del tipo $p(x) \propto x^{-\alpha}$, se ha hecho uso del *toolbox* de Matlab propuesto por Clauset et al. (2009).

En la práctica, muy pocos fenómenos cumplen empíricamente la ley de potencias para todos los valores de x. Generalmente esta ley se aplica tan sólo para valores mayores a un cierto valor x_{min}. En estos casos se dice que la cola de la distribución describe la ley de potencias («power-law»).

A continuación vamos a intentar modelar la distribución de las peticiones según una ley de potencias. Para ello, considérese la distribución continua

$$p(x) = \frac{\alpha - 1}{x_{min}} \left(\frac{x}{x_{min}} \right)^{-\alpha} \tag{5.2}$$

El estimador de máxima verosimilitud (MLE - *Maximum Likelihood Estimator*) para el exponente α es:

$$\widehat{\alpha} = 1 + n \left[\sum_{i=1}^{n} ln \frac{x_i}{x_{min}} \right] \tag{5.3}$$

donde x_i, con $i = 1, \ldots, n$ son observaciones independientes de la variable tales que $x_i > x_{min}$. El límite inferior x_{min}, que determina el intervalo en el que se cumple la ley de potencias, se calcula según el procedimiento de Kolmogorov-Smirnov (Massey, 1951).

En muchos casos resulta de utilidad considerar la distribución acumulativa complementaria (CCDF - *Complementary Cumulative Distribution Function*) de la variable, digamos $P(x)$, definida como $P(x) = Pr(X \geq x)$:

$$P(x) = \int_{x}^{\infty} p(x)dx = \left(\frac{x}{x_{min}} \right)^{1-\alpha} \tag{5.4}$$

En la Figura 5.3 se muestra la función CCDF de peticiones a cada uno de los servicios de mapas estudiados, junto con los valores de x_{min} y α estimados. Como se puede observar la función CCDF obtenida se ajusta bastante bien a una ley de potencias, en el caso de Cartociudad e IDEE-Base, si bien el comportamiento observado en PNOA es muy diferente. Este hecho se justifica por la anómala concentración de peticiones en las escalas de mayor resolución de detalle, en las que las peticiones son más dispersas.

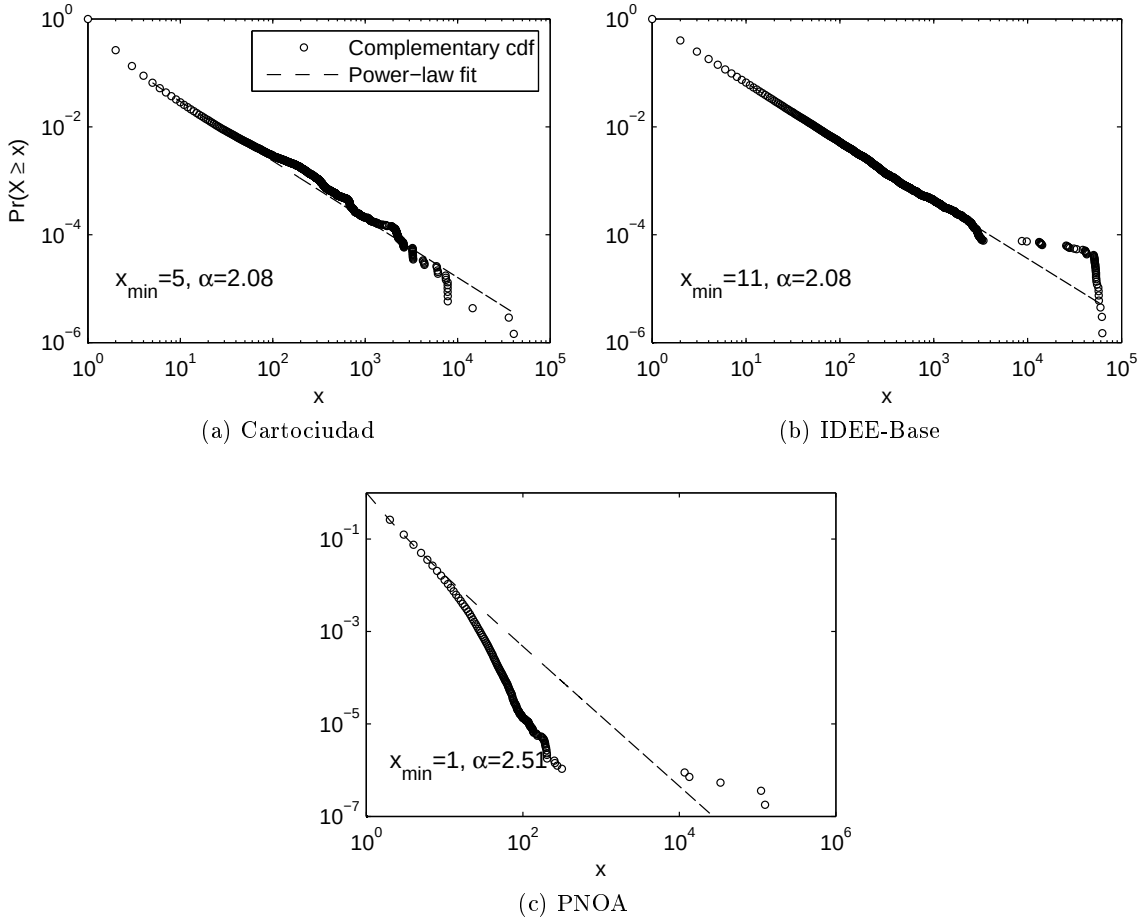
Dentro del ámbito de los servicios de mapas, existen otros indicios de la aparición de este tipo de distribuciones en la frecuencia de acceso de las teselas (Li et al., 2012; Fisher, 2007b; Wang et al., 2010).

5.6. Distribución de peticiones por escala

En la Figura 5.4 se recoge la distribución de peticiones realizadas por los usuarios sobre las diferentes escalas. En los servicios IDEE-Base y Cartociudad se observa un pico de peticiones realizadas sobre la escala 4. Se ha comprobado que este nivel de resolución es el más adecuado para mostrar por completo el territorio nacional en una única pantalla de mapa. Por tanto, esta escala de representación resulta idónea para la carga inicial del mapa que se presenta a los usuarios en un servicio de mapas de ámbito nacional, como ocurre en el caso del visor del IGN[5]. De igual forma, este es el nivel de resolución escogido por Google Maps España[6] en su mapa de inicio. Este hecho ofrece una justificación razonable sobre la existencia de estos máximos localizados en esta escala.

Sin embargo, exceptuando el pico anómalo de carga detectado en esta escala, los niveles de resolución con mayor nivel de detalle reciben un mayor número de peticiones que los de resolución más baja. Esto difiere de lo ocurrido en otros escenarios en los que se utilizan

[5] http://www.cartociudad.es/visor
[6] http://maps.google.es



(a) Cartociudad

(b) IDEE-Base

(c) PNOA

Figura 5.3: Función de distribución acumulativa complementaria (CCDF) de referencias a las teselas y su ajuste por ley de potencias.

imágenes multi-resolución, como en (Talagala et al., 2000), donde las escalas más bajas reciben un mayor número de peticiones. En este último trabajo se gestionan imágenes artísticas de un museo, utilizando una técnica de teselado multi-resolución similar al usado en el ámbito de los mapas.

Sin embargo, dado que el número de teselas crece exponencialmente con el nivel de resolución, la densidad de peticiones (número de peticiones recibidas por cada tesela presente en el espacio de teselado) es mayor en las escalas más bajas.

Por otra parte, cabe destacar que la mayor parte de las peticiones recibidas por el servicio PNOA se concentran en los dos niveles de mayor resolución (escalas 18 y 19). Esta anomalía se produce debido a que en el visor del IGN la capa de ortofotos del PNOA tan sólo está activa para estas escalas.

5.7. Distribución del Tamaño de las Teselas

A partir de los registros de acceso de los servicios analizados puede extraerse el número de *bytes* transferidos para cada tesela solicitada, no su tamaño real. Tal y como se expone en (Arlitt y Williamson, 1997), ambos valores pueden diferir. Por ejemplo, en la mayor

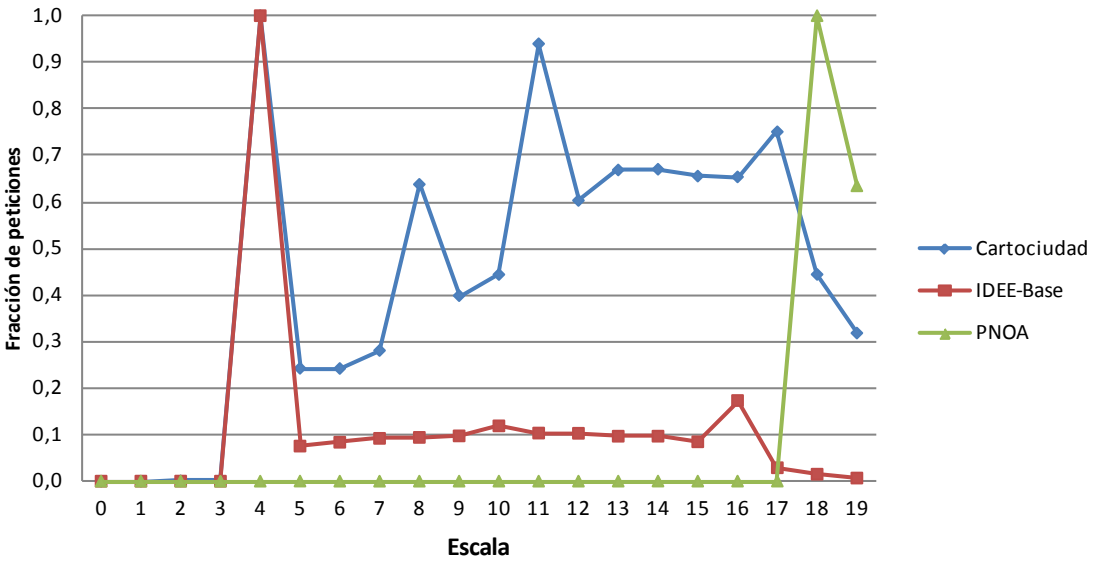


Figura 5.4: Distribución normalizada de las peticiones en función del nivel de resolución para los diversos servicios bajo estudio.

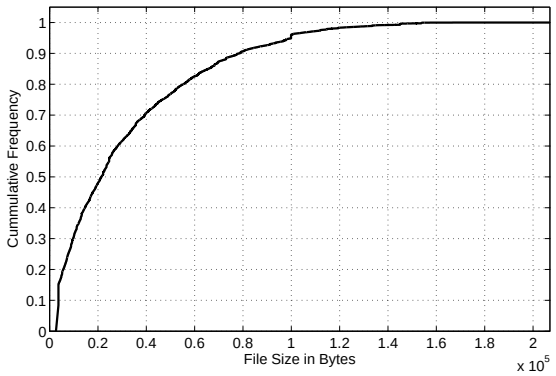
parte de los navegadores Web, los usuarios pueden abortar la transferencia de un documento en cualquier momento, haciendo que el tamaño transferido recogido en los *logs* pueda ser inferior que el tamaño real del documento. Sin embargo, dado el reducido tamaño de los objetos gestionados por los servicios de mapas analizados (inferior a 100KB) es poco probable que se produzca una interrupción en la transferencia de las teselas, por lo que la desviación del valor recogido en los *logs* respecto al valor real se considera despreciable.

En la Figura 5.5 se muestra la frecuencia acumulada en función del tamaño de los objetos solicitados (fracción del número de peticiones recibidas de objetos cuyo tamaño es menor o igual que el valor indicado en el eje horizontal). Así, en el caso de Cartociudad, el 70 % de las peticiones recibidas corresponde a objetos cuyo tamaño no excede de 40KB.

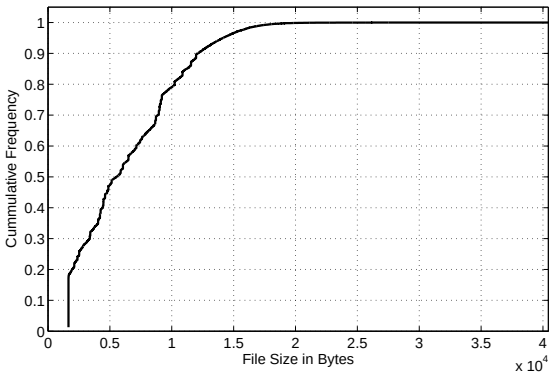
En Cartociudad (Figura 5.5a) e IDEE-Base (Figura 5.5b) se aprecia un subconjunto de teselas de pequeño tamaño (3657 bytes y 1652 bytes, respectivamente) que reciben un elevado número de peticiones. Estos conjuntos de teselas corresponden a imágenes uni-color, sin texturas adicionales, y que ocupan el mismo tamaño de almacenamiento en disco. Sin embargo, este escalón no se produce en PNOA, debido a que maneja teselas ráster de ortofotos aéreas, a diferencia de los otros servicios, que manejan datos vectoriales rasterizados. En una ortofoto es poco probable obtener teselas mono-color como en el caso de teselas obtenidas a partir de datos vectoriales.

En un servicio Web convencional, el tamaño de los objetos servidos presenta una gran dispersión, a menudo de varios órdenes de magnitud. Esto se debe a la gran heterogeneidad en el tipo de objetos que gestiona (Arlitt y Williamson, 1996): desde un documento de texto de unos pocos *bytes* a un archivo de vídeo de varios *MegaBytes*.

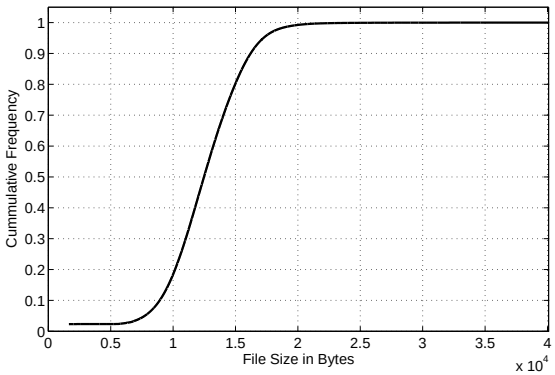
Sin embargo, en el caso de los servicios de mapas teselados, todos los objetos que gestiona (tan sólo se consideran las peticiones *GetMap*) son imágenes de mapa de tamaño uniforme (típicamente 256 x 256 píxels), por lo que la variabilidad de tamaños es mucho menor.



(a) Cartociudad (escala 8)



(b) IDEE-Base (escala 8)



(c) PNOA (escala 18)

Figura 5.5: Distribución del tamaño de las teselas para los diferentes servicios analizados.

Una medida de esta dispersión es el CoV (*Coefficient of Variation*), que se define como la relación entre la desviación típica de una muestra (σ) y su media (μ):

$$CoV = \frac{\sigma}{\mu} \tag{5.5}$$

Así, mientras que Arlitt y Williamson (1996) encontraron valores de CoV comprendidos entre 3.45 y 11.19 durante el análisis de 6 servidores Web tradicionales, los valores encontrados para los tres servicios de mapas analizados son de tan sólo 0.975 para Cartociudad, 0.655 en IDEE-Base y 0.258 en el caso de PNOA.

En el caso del servicio de mapas del PNOA, la mayor parte de las teselas solicitadas se encuentran en el rango de 6 a 20 KB. Para los otros dos servicios, el 90 % de las peticiones se realiza sobre teselas con tamaños comprendidos entre [3.6 - 80] KB en el caso de Cartociudad y [1.6 - 12] KB para IDEE-Base.

Por otra parte, cabe preguntarse si existe algún tipo de correlación entre la frecuencia de accesos a un documento y su tamaño, puesto que su respuesta puede afectar al diseño de los algoritmos de caché.

Para analizar esta dependencia, Breslau et al. (1999) calcularon el coeficiente de correlación entre la frecuencia de accesos y el tamaño de los documentos a partir de los registros de peticiones de 6 *proxies* pertenencientes a instituciones académicas, corpora-

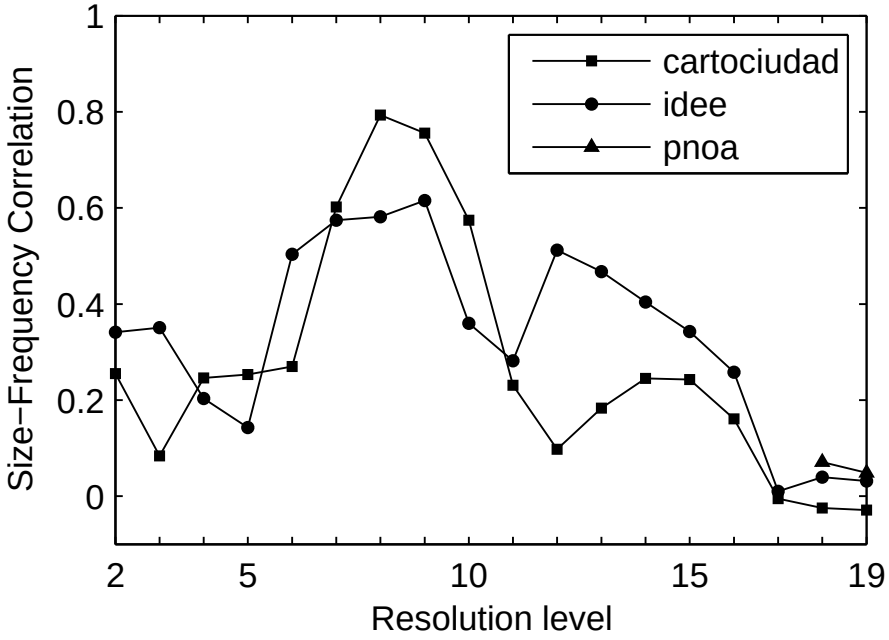


Figura 5.6: Correlación entre la frecuencia de acceso a las teselas y su tamaño para cada escala de representación.

ciones e ISP (*Internet Service Provider*). Los resultados de dicho estudio arrojaron que la correlación existente era despreciable (valores dentro del intervalo [-0.08 , 0.04]).

De forma similar, en el presente trabajo se ha calculado el coeficiente de correlación a partir de los registros de los servidores analizados. Los coeficientes de correlación globales, utilizando todos los registros de cada servicio, son despreciables. Al analizar este resultado debe tenerse en cuenta que la interacción de los usuarios con el mapa depende en gran medida de la escala de representación y de la capa de mapa a visualizar (parámetro *layers* de WMS). Asimismo, el tamaño medio de las teselas, entendido como el espacio de almacenamiento en disco ocupado por éstas, varía significativamente en función de estos dos parámetros. Por este motivo, se han repetido los cálculos de correlación para cada uno de los distintos servicios, para una capa concreta y para las distintas escalas de representación.

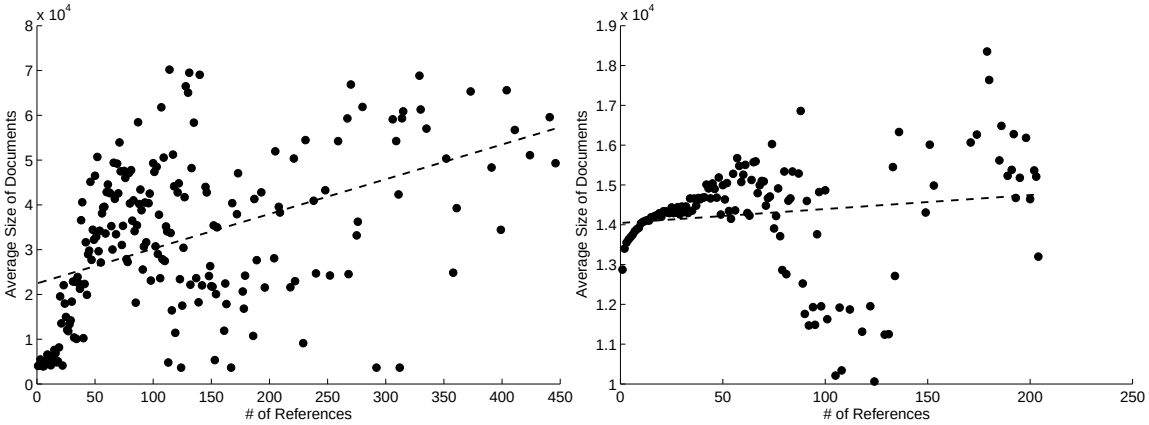
Los coeficientes de correlación entre las dos variables (digamos X e Y) se han calculado según la siguiente expresión:

$$\rho_{XY} = \frac{\sum_i (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_i (X_i - \overline{X})^2 \sum_i (Y_i - \overline{Y})^2}} \tag{5.6}$$

El valor de este coeficiente de correlación ρ_{XY} varía en el intervalo [-1,1] y sus valores se interpretan de la forma siguiente:

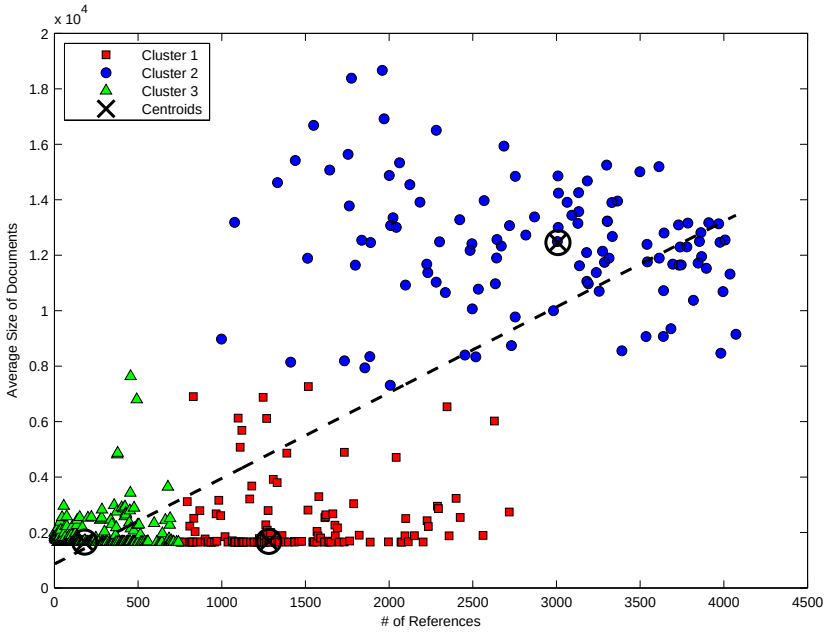
- Si $\rho_{XY} = 1$, existe una correlación positiva perfecta. El coeficiente indica una dependencia total entre las dos variables denominada relación directa: cuando una de ellas aumenta, la otra también lo hace en proporción constante.
- Si $0 < \rho_{XY} < 1$, existe una correlación positiva.

- Si $\rho_{XY} = 0$, no existe relación lineal. Pero esto no necesariamente implica que las variables sean independientes: pueden existir todavía relaciones no lineales entre las dos variables.
- Si $-1 < \rho_{XY} < 0$, existe una correlación negativa.
- Si $\rho_{XY} = -1$, existe una correlación negativa perfecta. El coeficiente indica una dependencia total entre las dos variables llamada relación inversa: cuando una de ellas aumenta, la otra disminuye en proporción constante.



(a) Cartociudad, nivel 9

(b) PNOA, nivel 18



(c) IDEE-Base, nivel 8

Figura 5.7: Tamaño medio de los objetos frente a su frecuencia de referencia.

Los resultados, calculados según la Ecuación 5.6, se muestran en la Figura 5.6. Como

se puede observar, se obtienen coeficientes de correlación elevados para múltiples escalas. Estos coeficientes indican la presencia de una dependencia espacial positiva significativa entre la popularidad de las teselas y su tamaño.

Estos resultados son coherentes con los obtenidos en (Quinn y Gahegan, 2010), donde se llegó a esta misma conclusión, aunque de forma indirecta: cacheando las teselas que su modelo estimaba como de mayor popularidad, el porcentaje de ahorro en términos de número de teselas excedía al medido en términos de consumo de espacio de almacenamiento en disco.

El hecho de que las teselas de mayor "interés" (las más solicitadas) incurran en un mayor coste de almacenamiento que aquellas omitidas se interpreta de la forma siguiente. Generalmente, las zonas "populares" presentan una mayor variación cromática y de patrones, por lo que estas zonas populares que se introducen en caché almacenadas como imágenes comprimidas en formatos como JPEG o PNG, usan una mayor proporción del espacio en disco que las teselas no cacheadas que presentan pocos contrastes en la imagen.

Estos resultados son sin embargo divergentes con los obtenidos por Breslau et al. (1999) durante un estudio de tráfico Web convencional, en el que ambas variables, frecuencia de accesos y tamaño de tesela, eran estadísticamente independientes. Asimismo, los resultados son contradictorios con los obtenidos en (Cunha et al., 1995), de nuevo en un servicio Web tradicional.

Para respaldar los resultados obtenidos, en la Figura 5.7 se muestra el tamaño medio de las teselas que son pedidas exactamente n veces en los *logs*, mostrando n en el eje horizontal. Tal y como muestran los coeficientes de correlación, se aprecia una mayor dependencia espacial entre la frecuencia de accesos y el tamaño de las teselas en el servicio IDEE-Base. Sobre los datos se ha aplicado un *cluster* de tipo K-Means (Seber, 1984). El cluster 3 corresponde a objetos de pequeño tamaño y cuya frecuencia de accesos es baja. El cluster 2 corresponde a teselas de gran tamaño que se piden mucho. El cluster 1 son excepciones que empeoran la dependencia lineal entre ambas variables. Comparando estas figuras con las análogas presentadas por Breslau et al. (1999), se observan que la dependencia entre el tamaño de los objetos y su popularidad es mucho más significativa en un servicio de mapas que en el caso de un servicio Web convencional.

La presencia de dependencia entre ambas variables es un importante hallazgo que puede ser explotado para el diseño de nuevas y más eficientes políticas de gestión de una caché de mapas. Esto se pone de manifiesto en la estrategia de reemplazo basada en redes neuronales presentada en el Capítulo 6.1, donde el tamaño de los objetos se utiliza como una entrada del modelo predictivo.

5.8. Análisis temporal

La distribución temporal de las peticiones es un aspecto que no conviene obviar, y debe tenerse en cuenta para la gestión y optimización de las cachés espaciales. La detección de picos de carga (número elevado de peticiones simultáneas) puede utilizarse para la caracterización de la calidad de servicio. En presencia de estos picos, la QoS será menor que en el caso de que se reciba un menor número de peticiones simultáneas. Por otra parte, los intervalos de tiempo con baja actividad son buenos candidatos para realizar labores de mantenimiento y actualización del servicio de caché.

Para la realización de este análisis se han llevado a cabo los siguientes estudios:

- Carga de trabajo media de cada servicio a lo largo del día y para cada día de la semana. Se obtiene como la media de todas las semanas registradas en los *logs* de dichos servicios (ver Figura 5.8).
- Carga de trabajo media a lo largo del día, realizada haciendo un promedio de todos los días de la semana (ver Figura 5.9).
- Carga de trabajo para cada día de la semana (ver Figura 5.10).

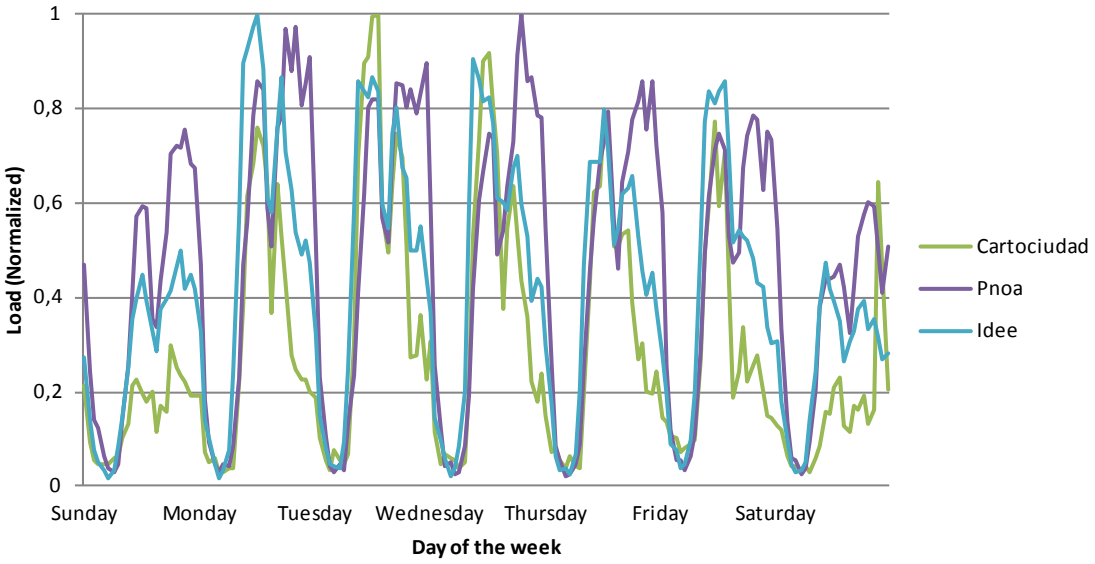


Figura 5.8: Distribución de las peticiones de los usuarios en función del día de la semana y la hora.

A partir del análisis anterior se han extraído las siguientes observaciones:

- La posición de los máximos y mínimos relativos es muy similar durante todos los días de la semana (ver Figura 5.8).
- La carga durante los fines de semana es significativamente más reducida que durante los días de diario (ver Figura 5.10).
- La carga se reduce drásticamente durante las horas nocturnas, alcanzando un mínimo absoluto en torno a las 05:00 h con valores de carga por debajo del 8 % de la máxima alcanzada durante todo el día (ver Figura 5.9). Es de esperar que, en un servicio más global, como *Google Maps*, la carga fuese más uniforme a lo largo del día debido a la diferencia horaria entre las posiciones de los clientes.
- La carga de trabajo es mayor durante las partes centrales del día, con un mínimo relativo situado en torno a las 15:00 h (ver Figura 5.9).

Estos resultados son consistentes con los obtenidos por Kefaloukos et al. (2012) al estudiar los patrones de acceso al servicio de mapas "Digital Map Supply" de Dinamarca. En

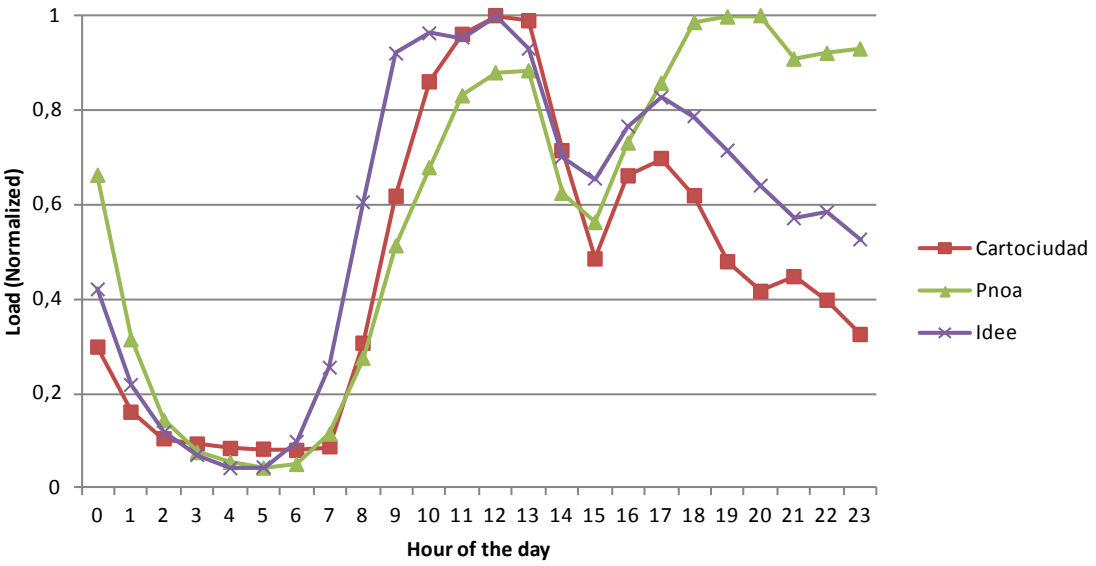


Figura 5.9: Distribución de las peticiones de los usuarios en función de la hora del día.

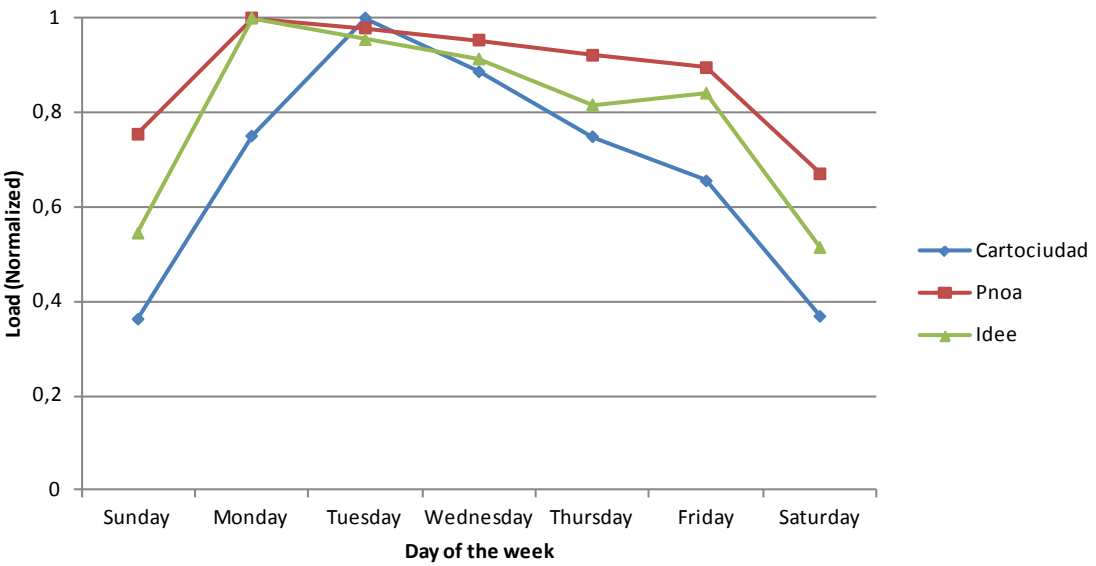
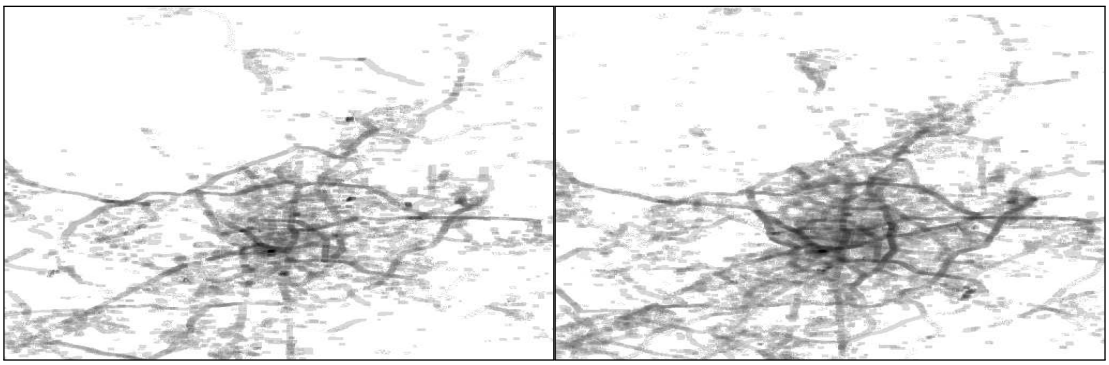


Figura 5.10: Distribución de las peticiones de los usuarios en función del día de la semana.

la Figura 5.11 se muestra el *heatmap* de peticiones recibidas en el servicio PNOA, distinguiendo por una parte las recibidas entre semana, y por otra parte las recibidas los fines de semana. Se observa que la distribución espacial de las peticiones es parecida, pero se aprecian múltiples diferencias. Por tanto, no sólo cambia el número de peticiones recibidas, sino también su distribución espacial.



(a) Viernes - Domingo (b) Lunes - Jueves

Figura 5.11: Estacionariedad de las peticiones en PNOA. Madrid capital.

5.9. Tiempos entre llamadas sucesivas de un mismo objeto

El siguiente análisis realizado se centra en los tiempos entre referencias sucesivas para los objetos que se piden en más de una ocasión. Estos tiempos han sido calculados para cada tesela, y posteriormente se han combinado para formar la distribución acumulativa de tiempos que se muestra en la Figura 5.12.

Tabla 5.3: Tiempos medios entre referencias sucesivas de un mismo objeto.

CARTOCIUDAD	IDEE_BASE	PNOA
6.8 días	1.3 días	12.1 días

En dicha figura se puede observar, por ejemplo, que en el servicio IDEE-Base se recibe un elevado número de peticiones con un tiempo entre llamadas sucesivas inferior al minuto. Este es el servicio con una menor diferencia de tiempos entre llegadas sucesivas de peticiones de un mismo objeto, tal y como se muestra en la Tabla 5.3. En el lado opuesto, y con una media de 12.1 días, el servicio PNOA es el que presenta un mayor tiempo entre llegadas sucesivas.

Conocer el tiempo entre peticiones sucesivas de un mismo objeto puede resultar beneficioso para el diseño de adecuadas políticas de gestión de caché. Así, en servicios que actualizan su cartografía con mucha frecuencia podría darse el caso de que el tiempo de vida de los objetos sea inferior al tiempo entre referencias. En este caso, el consumo de recursos al almacenar una tesela en la caché sería inútil, puesto que al cachear un objeto, la siguiente referencia al mismo se produciría una vez que éste hubiese expirado con alta probabilidad.

A continuación se ha llevado a cabo un análisis equivalente, pero considerando en este caso no el tiempo entre peticiones sino el orden de llegada de las mismas. En este caso, la componente de interés es la distribución de probabilidad $d(k)$ de, recibida una petición en el puesto i, la siguiente petición de este mismo objeto esté k posiciones después (es decir, que la petición del objeto i esté seguido de $k-1$ peticiones de otros objetos distintos del objeto i, seguido de otra petición del objeto i). Los resultados de este análisis se muestran en la Figura 5.13.

La distribución de los puestos en que se piden los objetos afecta directamente al ren-

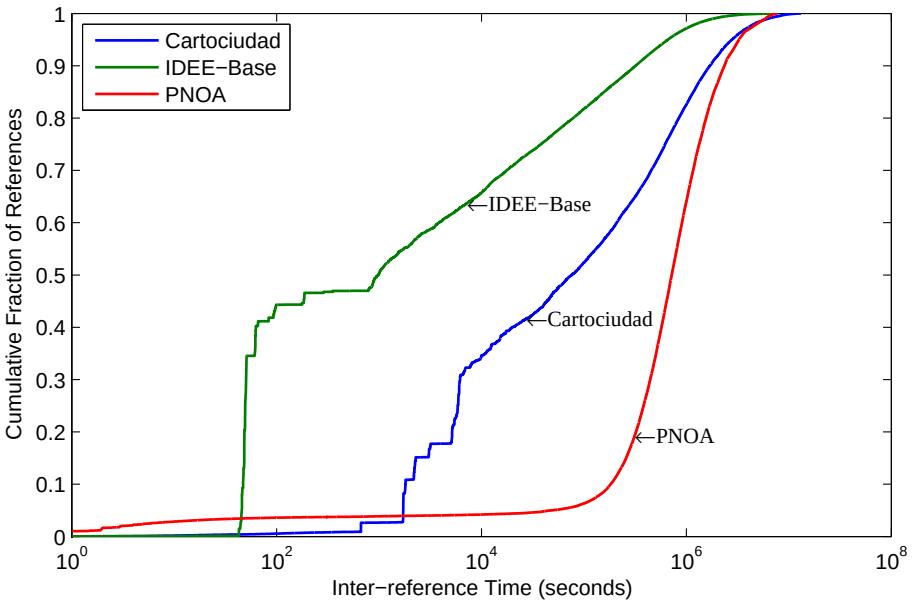


Figura 5.12: Distribución de tiempos entre peticiones sucesivas de un mismo objeto.

dimiento de diversos algoritmos de caché. Así, por ejemplo, en servicios que presentan una elevada localidad temporal en sus peticiones, el algoritmo LRU ofrece mejores resultados que la estrategia LFU (Breslau et al., 1999).

5.10. Autocorrelación espacial

En muchas aplicaciones que manejan datos espaciales, los sucesos ocurridos en una localización están influenciados en gran medida por los que ocurren en localizaciones vecinas (Shekhar y Xiong, 2008). La inclinación natural de que una variable muestre valores semejantes en función de la distancia entre los puntos en que es medida se conoce como dependencia espacial. Para medir esta dependencia espacial se utiliza la autocorrelación espacial. Si la variable presenta un patrón sistemático en su distribución espacial se dice que ésta es espacialmente autocorrelada. La autocorrelación espacial puede ser positiva, negativa o nula. La autocorrelación espacial positiva indica que valores o propiedades semejantes tienden a situarse cercanos. La negativa indica que valores o propiedades diferentes tienden a estar unos cerca de otros. Una autocorrelación espacial nula indica que los patrones son independientes, ya que los datos aleatorios independientes e idénticamente distribuidos (iid) son invariantes respecto a su localización espacial.

Dada una variable X, con valores X_i, X_j en las respectivas localizaciones i y j, la función buscada debe satisfacer las siguientes condiciones (De Smith et al., 2007):

1. Si el par (X_i, X_j) son del mismo signo, $f(X_i, X_j) > 0$

2. Si el par (X_i, X_j) son de signos opuestos, $f(X_i, X_j) < 0$

3. Si el par (X_i, X_j) son ambos de gran valor, $f(X_i, X_j)$ es grande

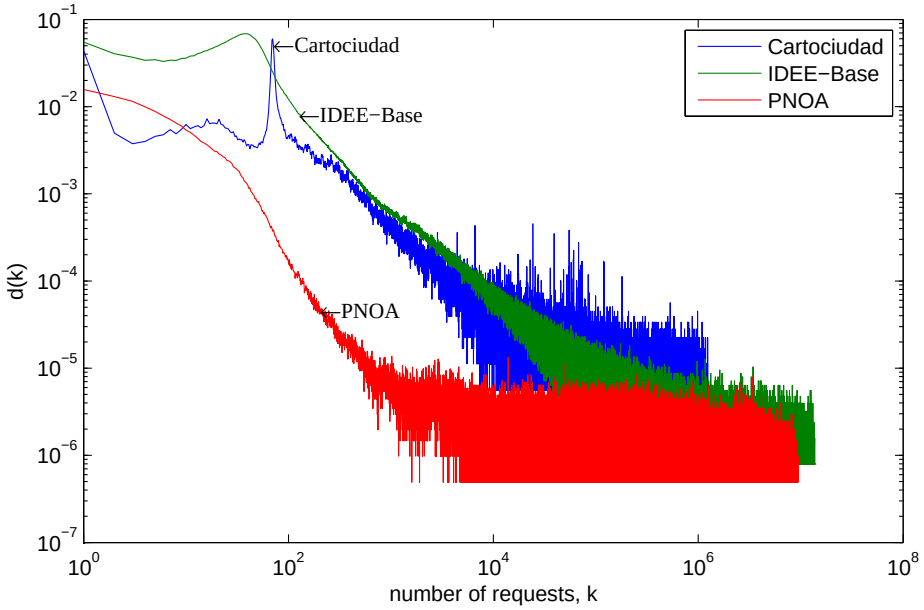


Figura 5.13: Distribución de probabilidad d(k) de recibir otra petición del mismo objeto k peticiones después.

4. Los patrones de adyacencia, reflejados en la matriz W, deben tenerse en cuenta en la función

La forma más sencilla de conseguir lo anterior consiste en multiplicar los valores de las zonas adyacentes

$$\sum_i \sum_j W_{i,j} X_i X_j \tag{5.7}$$

o ajustando los valores respecto a la media para obtener momentos estadísticos centrales

$$\sum_i \sum_j W_{i,j} (X_i - \overline{X})(X_j - \overline{X}) \tag{5.8}$$

Asumiendo que los pesos $W_{i,j}$ son binarios, éstos sólo identifican qué elementos deben de incluirse o excluirse del cálculo. Puede realizarse un ajuste por el número de elementos incluidos en el sumatorio, y producir un valor de covarianza

$$\frac{\sum_i \sum_j W_{i,j}(X_i - \overline{X})(X_j - \overline{X})}{\sum_i \sum_j W_{i,j}} \tag{5.9}$$

Finalmente, para normalizar esta expresión de covarianza hay que dividir por la varianza de los datos $\frac{\sum_i (X_i - \overline{X})^2}{n}$

$$I = \frac{n \sum_i \sum_j W_{i,j}(X_i - \overline{X})(X_j - \overline{X})}{(\sum_i \sum_j W_{i,j}) \sum_i (X_i - \overline{X})^2} \tag{5.10}$$

La ecuación anterior corresponde al índice I de Moran, uno de los métodos clásicos para calcular la autocorrelación espacial, y el cuál sigue siendo la técnica habitual para

calcularla. En notación matricial puede expresarse de la forma siguiente:

$$I = \frac{zWz^t}{zz^t} \tag{5.11}$$

donde $z = (x_1 - \overline{x}, x_2 - \overline{x}, \ldots, x_n - \overline{x})$, z^t es la matriz traspuesta de z, y W es la matriz de contigüidad de tamaño $n \times n$.

Hay que resaltar que el índice I expresado en (5.10) no depende únicamente de la variable X, sino también de la disposición espacial de los datos. Ésta se refleja a través de la matriz de adyacencia, W. Si la localización i es adyacente a la localización j, entonces esta disposición espacial recibe un peso de 1; en caso contrario el peso es 0. En este caso la matriz de adyacencia recibe el nombre de matriz de contigüidad. Otra opción para calcular la matriz de adyacencia es definir los pesos como la inversa del cuadrado de la distancia $(1/d_{i,j}^2)$ entre los puntos i y j. Los métodos anteriores se ajustan bien para una red sencilla de celdas, pero no para reflejar la interacción espacial entre polígonos complejos e irregulares. En (Yang et al., 2008) se computa la matriz de adyacencia en base al borde común entre polígonos adyacentes, como se ilustra en (5.12).

$$W_{i,j} = \beta_{i,j}/P_i \tag{5.12}$$

donde $\beta_{i,j}$ es la longitud del borde compartido entre las celdas i y j, y P_i es el perímetro de la celda i.

El índice I de Moran (5.10) da una idea general del comportamiento espacial de la variable bajo estudio, lo que se denomina como *Autocorrelación Espacial Global*. A diferencia de ésta, la *Autocorrelación Espacial Local* es capaz de indicar agrupamientos espaciales significativos para cada localización e identificar puntos calientes. De los múltiples Indicadores Locales de Asociación Espacial (LISA - *Local Indicators of Spatial Association*) (Anselin, 1995) se destaca el Índice Local de Moran. Se trata de la versión local de (5.10), donde el índice se descompone en la contribución individual de cada observación, como se aprecia en (5.13).

$$I_i = \frac{n(X_i - \overline{X}) \sum_j W_{i,j}(X_j - \overline{X})}{(\sum_i \sum_j W_{i,j}) \sum_i (X_i - \overline{X})^2} \qquad I = \sum_i I_i \tag{5.13}$$

Otro procedimiento para cuantificar la autocorrelación espacial es el método de Geary (Geary, 1954). Este método difiere principalmente del método de Moran en que la interacción entre i y j no se mide como la desviación respecto a la media, sino por la diferencia de valores de cada observación. El coeficiente C de Geary se define como:

$$C = \frac{(n-1) \sum_i \sum_j W_{i,j}(X_i - X_j)^2}{2 \sum_i \sum_j W_{i,j}(X_i - \overline{X})^2} \tag{5.14}$$

Donde:

- $C < 1$ indica autocorrelación espacial positiva.
- $C = 1$ indica la inexistencia de autocorrelación espacial.
- $C > 1$ indica autocorrelación espacial negativa.

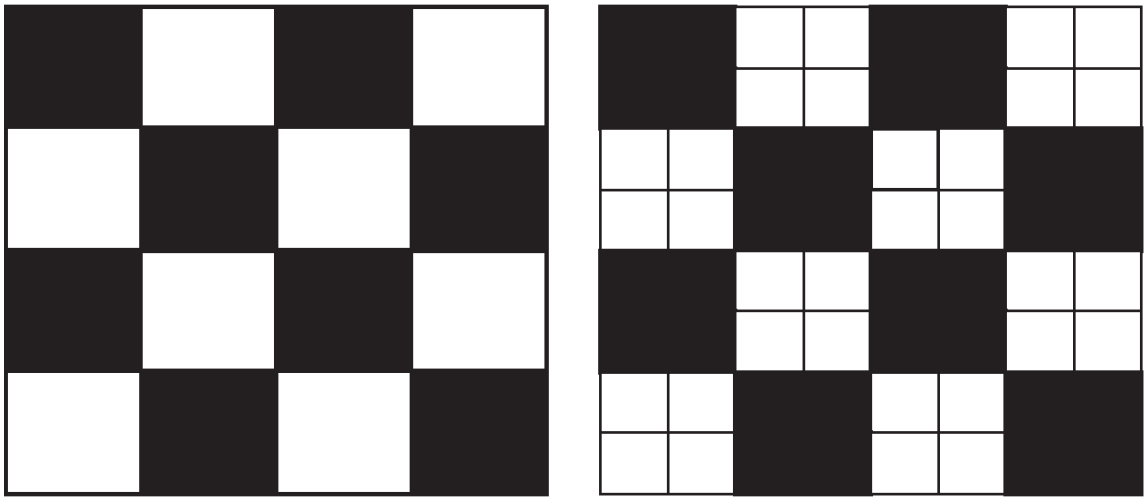


Figura 5.14: Dependencia de la autocorrelación espacial con la escala de representación. Misma cuadrícula para dos escalas de resolución sucesivas. a) raster de 4x4 (izda), b) raster de 8x8 (dcha). Fuente: (Shekhar y Xiong, 2008).

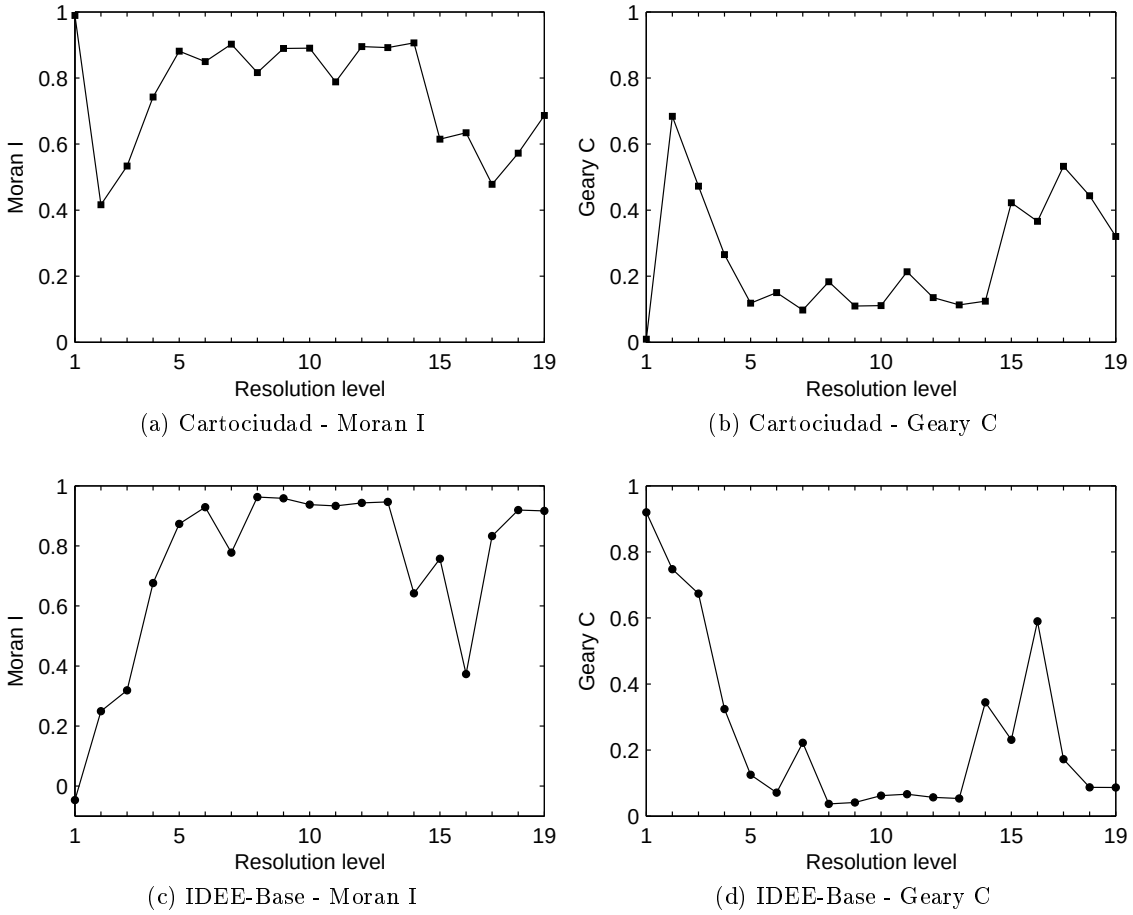
Existen algunos programas para el cálculo de las ecuaciones de Moran y Geary anteriores, entre los que destacan GeoDa[7], ArcGIS[8], `R`[9] y MATLAB[10], los dos últimos de carácter más general. Para la realización del presente estudio se ha hecho uso de `R`, y especialmente del paquete `spdep`[11]: *"Spatial dependence: weighting schemes, statistics and models"*.

La existencia de una fuerte autocorrelación espacial entre peticiones vecinas podría ser utilizada para desarrollar algoritmos de actualización de caché en tiempo real. Ante una petición realizada sobre una localización geográfica que presenta una elevada autocorrelación espacial, es de esperar que se realicen peticiones sobre localizaciones adyacentes a esta, por lo que se pueden tomar medidas para mejorar el tiempo de respuesta de las mismas.

Hay que destacar que, por lo general, el grado de autocorrelación espacial es dependiente de la escala o nivel de resolución espaciales, como se aprecia en la Figura 5.14. La red en la Figura 5.14.a presenta una alta autocorrelación espacial negativa, ya que cada celda tiene un color diferente a sus celdas vecinas. Dividiendo cada celda en cuatro y manteniendo la misma distribución (Figura 5.14.b), se obtiene una autocorrelación espacial positiva entre celdas blancas y negras. Este ejemplo artificial contradice la hipótesis del *principio de localidad*, por lo que tendremos que compaginar la posibilidad de esta aparente aleatoriedad de la correlación espacial entre niveles de resolución con la necesidad de implementar heurísticas que utilicen la información de otros niveles.

En la Figura 5.15 se muestran las medidas de autocorrelación global, obtenidas mediante el índice I de Moran y el coeficiente C de Geary, calculados a partir de los datos extraídos de los *logs* de Cartociudad e IDEE-Base, en función del nivel de resolución. Como se puede observar, la autocorrelación es dependiente de la escala de resolución, tal y como se anticipó anteriormente. Los elevados valores de autocorrelación espacial obtenidos

[7] http://geodacenter.asu.edu/
[8] http://www.arcgis.com/
[9] http://www.r-project.org/
[10] http://www.mathworks.es/
[11] http://cran.r-project.org/web/packages/spdep



(a) Cartociudad - Moran I

(b) Cartociudad - Geary C

(c) IDEE-Base - Moran I

(d) IDEE-Base - Geary C

Figura 5.15: Medidas de autocorrelación espacial de Moran&Geary de las peticiones de teselas para cada escala en los servicios WMS-C de Cartociudad e IDEE-Base.

respaldan analíticamente la idea intuitiva de dependencia espacial entre teselas vecinas, y ofrece la posibilidad de utilizar mecanismos para la carga dinámica de teselas adyacentes (ver Capítulo 8).

En la Figura 5.16 se muestra un correlograma de Moran para el servicio IDEE-Base al nivel de resolución 10. El correlograma es un gráfico que muestra cómo cambia la autocorrelación espacial, expresada mediante el índice I de Moran en este caso (en ordenadas) en función de la distancia (en abscisas) en términos de número de teselas según la matriz de adyacencia (Cliff y Ord, 1981). En este caso, se observa cómo la autocorrelación espacial decae rápidamente en función de la distancia. Esta información puede resultar de utilidad para la determinación del tamaño óptimo de *metatile* en una estrategia de carga heurística.

5.11. Correlación espacial cruzada

La correlación espacial cruzada mide el grado de dependencia espacial existente entre dos variables, digamos X, Y. Uno de los posibles indicadores de dependencia espacial entre variables es la ecuación bivariable de Moran (5.15), cuya interpretación es inmediata a

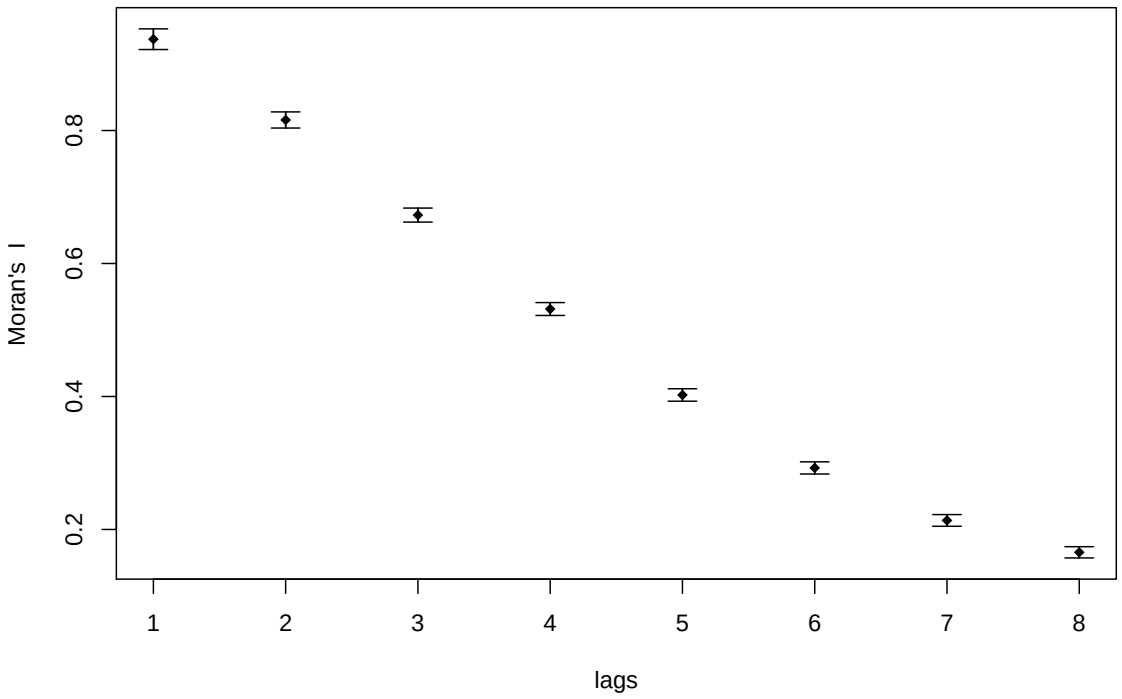


Figura 5.16: Correlograma de Moran para el servicio IDEE-Base, escala 10.

partir de su versión de una sola variable (5.10).

$$I_{XY} = \frac{n \sum_i \sum_j W_{i,j}(X_i - \overline{X})(Y_j - \overline{Y})}{(\sum_i \sum_j W_{i,j}) \sqrt{\sum_i (X_i - \overline{X})^2 \sum_j (Y_j - \overline{Y})^2}} \tag{5.15}$$

Al igual que se hizo para la autocorrelación, también se puede obtener la versión local de la correlación espacial cruzada de Moran, descomponiendo la expresión (5.15) como la contribución individual de cada observación (5.16).

$$I_{XYi} = \frac{n(X_i - \overline{X}) \sum_j W_{i,j}(Y_j - \overline{Y})}{(\sum_i \sum_j W_{i,j}) \sqrt{\sum_i (X_i - \overline{X})^2 \sum_j (Y_j - \overline{Y})^2}} \tag{5.16}$$

Para el cálculo de (5.15) se ha empleado Geoda, desarrollado en la Universidad de Arizona (USA). Éste permite, a partir de datos vectoriales almacenados en *shapefiles*, calcular distintos tipos de matrices de adyacencia (matriz de contigüidad, de contigüidad de mayor orden, en función de la distancia, de los K vecinos más cercanos, etc), y el cálculo de las correlaciones de una y dos variables, tanto globales como locales (Anselin et al., 2006).

Mediante el uso de la correlación bivaluada de Moran puede detectarse de forma analítica la existencia de *features* guiadoras de las peticiones de los usuarios, como una medida de interdependencia. En la Figura 5.17 se muestran los resultados obtenidos para dos *features* diferentes:

- Capa de hidrografía nacional[12].

[12]Obtenido de IDEE http://www.idee.es

- Datos de población de los municipios españoles en el año 2008[13].

El estudio ha sido particularizado para el área geográfica correspondiente a la Comunidad de Madrid, a las escalas indicadas. Como era de esperar, para el servicio de *Cartociudad*, la población de los municipios es una *feature* mucho más directora de peticiones que la correspondiente a los datos hidrográficos. La población municipal puede ser, por tanto, una *feature* candidata para el desarrollo de mecanismos avanzados de *seeding* y de carga dinámica para el servicio de caché WMS-C de *CartoCiudad*. A un nivel de detalle mayor, convendría estudiar elementos como los viales de distintas categorías. Esta técnica podría ser aplicable como función de aprendizaje para un procedimiento automatizado de detección de elementos directores de peticiones.

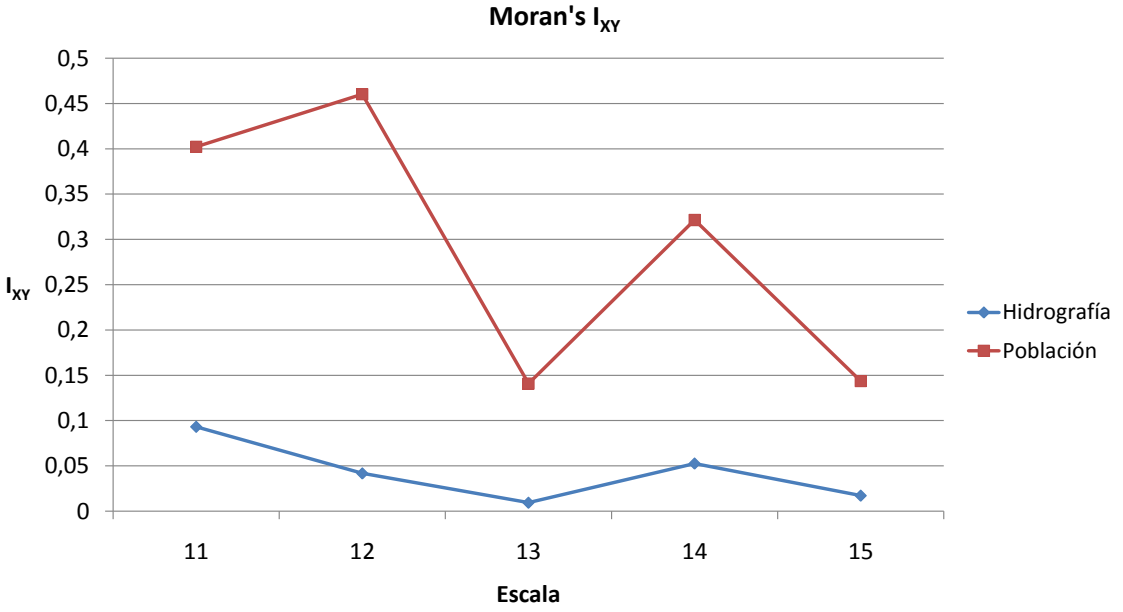


Figura 5.17: Correlación bivaluada de Moran.

Posteriormente, en el Capítulo 7.3 se proponen dos mecanismos para la precarga de teselas aprovechando la existencia de *features* directores de las peticiones de los usuarios.

5.12. Origen geográfico de las peticiones

Conocer el origen geográfico de los usuarios, y la relación entre éste y el contenido solicitado por los mismos, puede utilizarse para optimizar el funcionamiento de ciertos mecanismos de gestión de caché. Así, existen multitud de estudios (Kirchner et al., 2004; Cho, 2002; Fang et al., 2008) que aprovechan el conocimiento de la localización de los usuarios para anticipar el contenido que éstos van a solicitar y anticiparse realizando precarga de las teselas que se espera se van a realizar en un futuro cercano.

Por otra parte, este conocimiento puede utilizarse para conseguir un emplazamiento óptimo de los *datacenters* de replicación del servicio, en caso de haberlos.

[13]Obtenido del INE (Instituto Nacional de Estadística) http://www.ine.es/

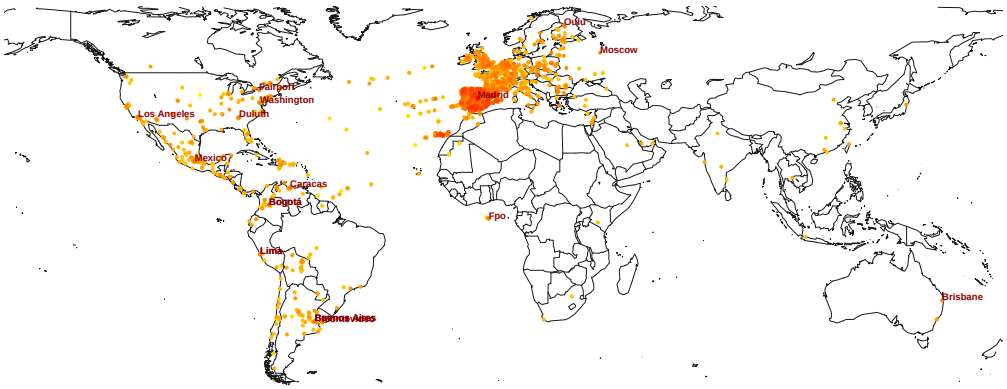


Figura 5.18: Posición geográfica de los clientes del servicio IDEE-Base obtenidas a partir de los logs de acceso.

A partir de los registros del servicio de mapas IDEE-Base se ha realizado una estimación del origen geográfico de los clientes. Para ello se ha hecho uso de las IPs registradas en los *logs* del servicio.

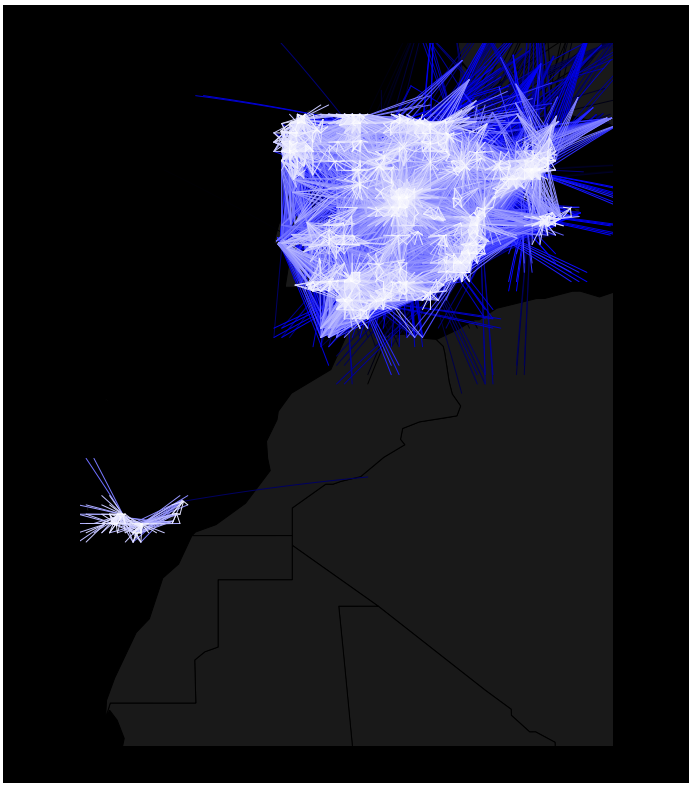


Figura 5.19: Relación entre la posición geográfica de los clientes del servicio IDEE-Base y el destino geográfico de sus peticiones.

Debido a la existencia de *proxies* entre el cliente y el servicio que registra las peticiones, es posible que la dirección recogida no sea la del cliente final sino la de un *proxy*. Para la obtención de la zona geográfica a partir de la IP se ha hecho uso del producto *GeoLite City* de la empresa MaxMind[14]. Se trata de una base de datos que geolocaliza direcciones IP y que se actualiza mensualmente. Se ha escogido la base de datos correspondiente a las fechas de recogida de los *logs* del servicio.

Sin embargo, hay que recalcar que las posiciones estimadas tienen una fiabilidad limitada. En (Poese et al., 2011) se analiza la granularidad de las entradas de diversas bases de datos para la geolocalización de IPs, incluída la utilizada aquí. Los resultados de dicho estudio indican que la precisión de estas bases de datos no suele llegar al nivel de ciudad esperado, sino que la estimación es más gruesa.

En la Figura 5.18 se muestra el origen geográfico estimado de las peticiones recibidas por el servicio IDEE-Base. Como se puede observar, la mayor parte de las peticiones se realizan desde España, seguida de Europa Occidental y Latino-América.

Por último, en la Figura 5.19 se muestra sobre un mapa las conexiones entre la posición geográfica (estimada) de los clientes y la zona geográfica solicitada en la petición de mapa. Este gráfico ha sido realizado utilizando las técnicas recomendadas por Paul Butler, empleado de Facebook, utilizadas para visualizar las conexiones entre los contactos en esta popular red social (Butler, 2010).

5.13. Aplicación al resto de la tesis

Algunas de las características del tráfico de los servicios de mapas analizados han servido como hipótesis para el desarrollo de diversos algoritmos de gestión de caché que se presentan en capítulos posteriores. En concreto, los elevados valores de autocorrelación espacial entre las peticiones de los usuarios han servido como hipótesis para el desarrollo de un modelo simplificado que agrupa las estadísticas recibidas entre teselas vecinas. Asimismo, la dependencia existente entre el espacio de almacenamiento de las teselas y su popularidad ha sido aprovechada en la estrategia de reemplazo basada en redes neuronales presentada en el Capítulo 6.1, donde el tamaño de los objetos se utiliza como una entrada más del modelo predictivo. Las características de estacionariedad de las peticiones han servido como hipótesis para el desarrollo del modelo descriptivo presentado en el Capítulo 7.2. Por otra parte, la presencia de *features* directoras de las peticiones de los usuarios ha sido explotada en el Capítulo 7.3 para el desarrollo de dos modelos predictivos para la precarga de teselas.

[14] http://dev.maxmind.com/geoip/geolite

Capítulo 6

Políticas de reemplazo de caché

Resumen: En el caso habitual de que no se disponga de espacio suficiente para poder albergar todos los objetos en la caché, cuando ésta se encuentra completa, un algoritmo de reemplazo debe determinar qué teselas deben ser reemplazadas. En este capítulo se proponen dos novedosas estrategias de reemplazo de caché específicas para servicios de mapas. La primera de ellas es una estrategia adaptativa basada en el uso de redes neuronales, que estima la popularidad de acceso futuro en base a ciertas propiedades de los objetos que gestiona: actualidad de referencia, frecuencia de referencia, y el tamaño de la tesela referenciada. La segunda, bautizada como *Spatial-LFU*, es una variante de la estrategia *Perfect-LFU*, simplificada aprovechando la correlación espacial existente entre las peticiones.

Los contenidos del presente capítulo han sido parcialmente publicados en (García et al., 2011f) y (García et al., 2011c).

La mayor parte de los servicios de mapas más populares consiguen una baja latencia de servicio debido a que cuentan con todas las imágenes pregeneradas en su caché. Sin embargo, los requisitos de almacenamiento que esta estrategia impone son generalmente prohibitivos para otros proveedores, forzando a los administradores a recurrir a cachés parciales que contengan tan solo un subconjunto de la cartografía. En este caso, cuando la caché se queda sin espacio para albergar nuevos objetos, un algoritmo de reemplazo debe determinar qué teselas deben ser reemplazadas.

Otro problema que debe ser tratado es el de mantener la consistencia en la caché, lo que se traduce en que las copias almacenadas en la misma deberían actualizarse cuando cambian las originales. Una aproximación común a este problema consiste en establecer un tiempo de vida (*Time-To-Live* - TTL) a los objetos almacenados. La caché considera que una copia almacenada está anticuada si su TTL ha expirado, de forma que los objetos obsoletos se eliminan de la misma. Aquellas teselas eliminadas por haber expirado su TTL, con alta probabilidad de ser pedidas a continuación, deberían solicitarse de nuevo al servidor de mapas remoto para recibir y almacenar una nueva copia. Esta aproximación es más eficiente cuando la cartografía se actualiza periódicamente en instantes conocidos de antemano, puesto que se evitan borrados innecesarios de caché.

6.1. Estrategia de reemplazo de caché mediante redes neuronales

En el Capítulo 4.4.3 se realizó un estudio del arte en estrategias de reemplazo de caché. En dicho estudio se recogen algunos trabajos en los que se aplican redes neuronales para el reemplazo de objetos en cachés Web convencionales. Sin embargo, la relevancia y novedad de este trabajo reside en el escenario en el que se aplica. No se ha encontrado en la literatura ningún estudio similar acerca de la aplicación de redes neuronales para la toma de decisiones de reemplazo en una caché de mapas.

A pesar de que la metodología subyacente ya ha sido tratada en trabajos relacionados sobre cachés Web, las distribuciones y atributos de las peticiones Web de mapas difieren en gran medida de aquellas propias de los servidores Web tradicionales. Como se mostró en el Capítulo 5, en este contexto, el tamaño del objeto solicitado está estrechamente relacionado con su "popularidad", por lo que puede usarse para estimar la probabilidad de accesos futuros. Las estrategias de reemplazo en las cachés Web tradicionales generalmente utilizan este parámetro tan sólo para evaluar el compromiso entre los aciertos conseguidos sobre un objeto y el espacio necesario para almacenarlo.

Las redes neuronales artificiales (ANN - *Artificial Neural Network*) se inspiran en la observación de que los sistemas biológicos de aprendizaje están compuestos de redes complejas de neuronas interconectadas. Del mismo modo, las redes neuronales artificiales se componen de un grupo denso de unidades interconectadas. Cada neurona artificial tiene como entradas un conjunto de números reales (en analogía a las dendritas presentes en los sistemas biológicos) y calcula una combinación lineal de las entradas (función de propagación). La suma es procesada a través de una función no-lineal, conocida como *función de activación* o *función de transferencia*, la cuál devuelve un único valor real, como se muestra en la Figura 6.1.

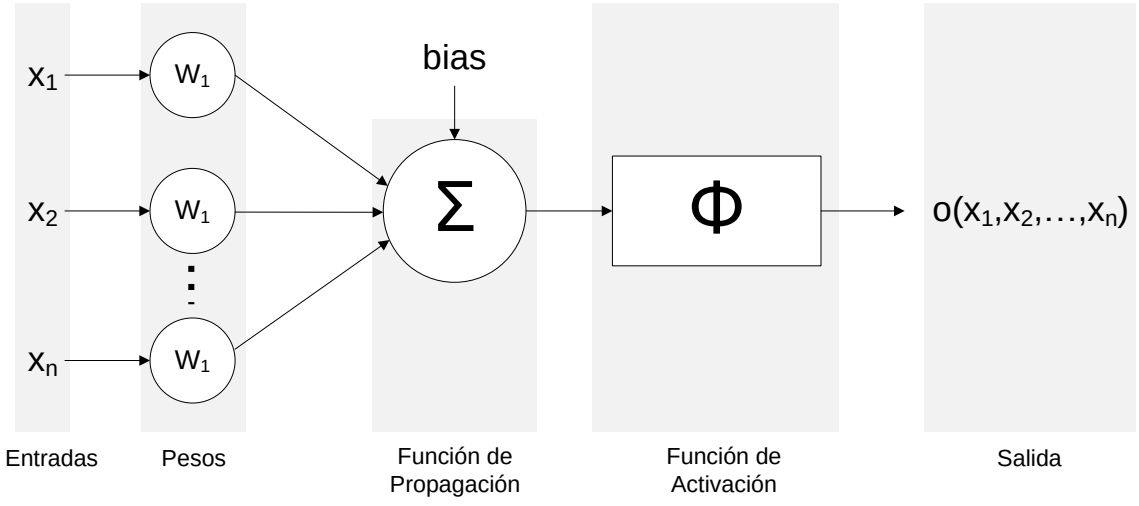


Figura 6.1: Neurona artificial.

En este trabajo se ha utilizado una clase especial de red neuronal conocida como "perceptrón multicapa" (MLP - *Multi-Layer Perceptron*), que es de tipo *feed-forward* (sin realimentación). Se trata de una arquitectura muy asentada y con multitud de implementaciones fiables y contrastadas. Como su nombre indica, estas redes están formadas por múltiples

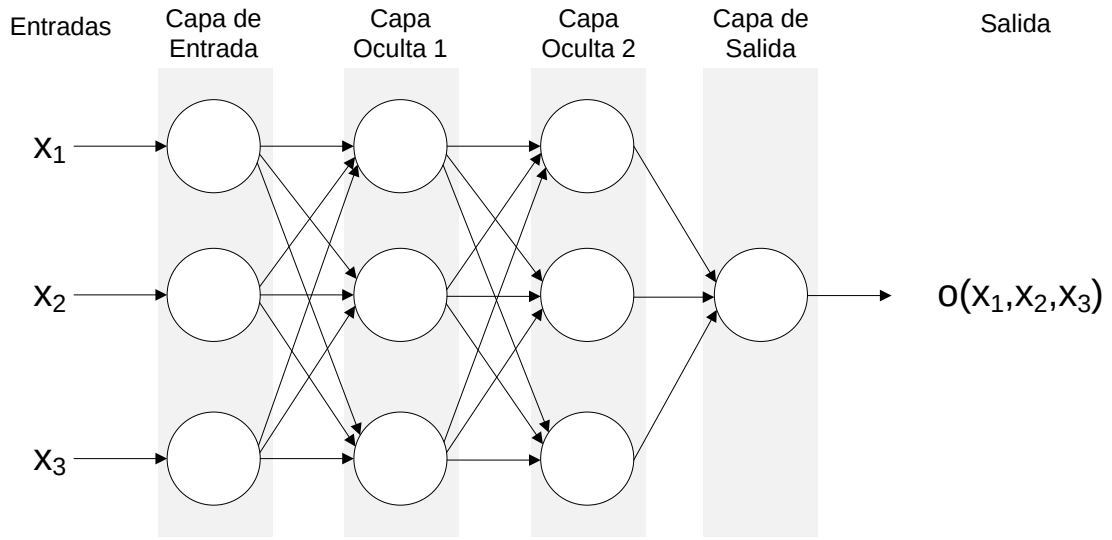


Figura 6.2: Red neuronal propuesta, de tipo *feed-forward* con dos capas ocultas.

capas, permitiendo resolver problemas que no son linealmente separables. Está formada por una capa de entrada compuesta por tres entradas, dos capas ocultas cada una compuesta por 3 nodos ocultos, y una única salida (Figura 6.2). Según la convención estándar, esta red puede etiquetarse como 3/3/3/1. Este tipo de redes compuestas por tres capas son capaces de aproximar cualquier función con precisión arbitraria (Cybenko, 1989).

Entrenar una neurona artificial consiste en elegir los valores de los pesos de forma que se obtenga la salida deseada para las entradas introducidas.

Los pesos de la red se han ajustado mediante entrenamiento supervisado usando los registros de peticiones descritos en el Capítulo 5, donde se conoce la salida para cada petición. Se ha utilizado el algoritmo «*backpropagation* con momento» para el entrenamiento de la red neuronal. En la Tabla 6.1 se recogen los parámetros principales utilizados en la red neuronal propuesta.

6.1.1. Entradas de la red neuronal

Como entradas de la red neuronal se utilizan tres propiedades de las peticiones Web: actualidad de referencia, frecuencia de referencia, y el tamaño de la tesela referenciada. Estas propiedades se emplean frecuentemente en los *proxy web cachés* para determinar la *cacheabilidad* de las peticiones (Venkataramani et al., 2002).

Las entradas han sido normalizadas de forma que todos los valores entren en el intervalo $[-1, 1]$. Para ello se ha hecho uso de un simple escalado lineal de los datos según la Ecuación (6.1), donde x e y son los valores de los datos antes y después de la normalización, respectivamente. x_{min} y x_{max} son los valores mínimo y máximo encontrados en los datos, y y_{max} y_{min} define el intervalo normalizado de forma que $y_{min} \leq y \leq y_{max}$. Esta normalización permite acelerar el aprendizaje de la red neuronal.

$$y = y_{min} + (y_{max} - y_{min}) \times \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6.1}$$

Los valores de actualidad de cada petición de tesela se computan como la cantidad de

Tabla 6.1: Parámetros de la red neuronal propuesta.

Parámetro	Valor
Arquitectura	Feed-forward Multilayer Perceptron
Capas ocultas	2
Neuronas en cada capa oculta	3
Entradas	3 (actualidad, frecuencia, tamaño)
Salida	1 (probabilidad de recibir una petición futura)
Funciones de activación	Log-sigmoidea en las capas ocultas, Tangente Hyperbólica en la capa de salida
Función de error	Error cuadrático medio (mse)
Algoritmo de entrenamiento	Backpropagation con *momentum*
Método de aprendizaje	Aprendizaje supervisado
Modo de actualización de los pesos	Batch mode
Learning rate	0.05
Momentum constant	0.2

tiempo desde que se recibió la última petición de ese objeto. Sin embargo, esta definición no es capaz de contemplar el caso en el que una tesela se pide por primera vez. Además, los valores así calculados podrían ser demasiado dispares como para reflejarse en una escala lineal.

Para resolver este problema, se ha utilizado una ventana deslizante en torno al instante de tiempo en el que se recibe cada petición, de la misma forma que se propone en (ElAarag y Romano, 2009b).

Mediante el uso de esta ventana deslizante, los valores de actualidad se calculan según la Ecuación (6.2).

$$recency = \begin{cases} max(SWL, \Delta T_i) & si\ el\ objeto\ i\ ya\ había\ sido\ pedido\ previamente \\ SWL & en\ otro\ caso \end{cases} \tag{6.2}$$

donde ΔT_i es el tiempo transcurrido desde la última petición de ese objeto, y SWL (*Sliding Window Length*) es la longitud de la ventana. Como norma general, la longitud de la ventana deslizante debería estar en torno al tiempo medio que un objeto permanece en la caché. Al igual que para la elección de los parámetros de la red neuronal, la selección del valor óptimo de SWL ha sido obtenido mediante experimentación. Tras probar distintas configuraciones, se ha seleccionado el valor $SWL = 24h$.

Los valores de actualidad calculados de esta forma son entonces normalizados según la Ecuación (6.1).

Los valores de frecuencia se calculan de la forma siguiente. Para una determinada petición, si se ha recibido una petición previa del mismo objeto dentro de la ventana, su valor de frecuencia se incrementa en una unidad. En caso contrario, este valor se divide entre el número de ventanas transcurridas desde su última petición. Una formalización de

lo anterior se recoge en la Ecuación (6.3).

$$frequency = \begin{cases} frequency + 1 & si\ \Delta T_i \leq SWL \\ MAX\left[\frac{frequency}{\frac{\Delta T_i}{SWL}}, 1\right] & en\ otro\ caso \end{cases} \tag{6.3}$$

La razón para el uso de este factor de decaimiento es que si la petición anterior se producía fuera de la ventana, entonces claramente el valor de salida de aquella sería 0, por lo que sería poco probable que permaneciese en caché (en este escenario se asume que el valor esperado del contador de frecuencia es proporcional al número de ventanas deslizantes que separa las peticiones). En cambio, si la petición previa se produjo dentro de la ventana, es probable que permaneciese en la caché de forma que su información frecuencial se mantendría por completo.

El tamaño de la tesela se extrae directamente a partir de los registros de las peticiones (*logs*) y se introduce como entrada de la red neuronal previa normalización. En contraposición a los *proxies* Web tradicionales, donde el tamaño de los objetos solicitados puede ser muy heterogéneo, en un servicio de mapas teselados todos los objetos corresponden a teselas de mapa que tienen las mismas dimensiones (típicamente de 256x256 píxels). Además, estas imágenes están normalmente codificadas en formatos eficientes como PNG, GIF o JPEG que hacen que raramente alcancen los 100 *KBytes*. En el Capítulo 5.7 se estudió la distribución del tamaño de las teselas de varios servicios.

Como se comenta en (Quinn y Gahegan, 2010), debido a una mayor variación cromática y de patrones, las áreas populares, almacenadas como imágenes comprimidas, utilizan una mayor proporción de espacio en disco que las teselas no solicitadas relativamente "vacías". En la Figura 7.8 (pág. 97) se ilustra lo anterior, comparando el espacio de almacenamiento para albergar una tesela "popular" correspondiente a un núcleo urbano, frente a una tesela "impopular" situada en el medio del mar, para cada uno de los servicios analizados.

Dada la dependencia entre el espacio de almacenamiento en disco de una tesela y su "popularidad", el tamaño puede resultar una entrada muy importante para que la red neuronal clasifique correctamente una tesela como *cacheable* o *no-cacheable*.

6.1.2. Salida de la red neuronal

Durante el proceso de entrenamiento, a cada registro correspondiente a la petición de una determinada tesela se le asocia una salida *booleana* (0 o 1) que indica si dicha tesela ha sido o no solicitada de nuevo dentro de la ventana, como se formaliza en la Ecuación 6.4.

$$target = \begin{cases} 1 & si\ se\ recibe\ de\ nuevo\ otra\ petición\ dentro\ de\ la\ ventana \\ 0 & en\ caso\ contrario \end{cases} \tag{6.4}$$

Una vez entrenada, la salida de la red neuronal será un número real en el rango [0,1] que debe interpretarse como la probabilidad de recibir una petición posterior del mismo objeto dentro de la ventana temporal.

Una petición se clasifica como *cacheable* si la salida de la red neuronal es mayor que 0.5. En caso contrario se clasifica como *no-cacheable*.

6.1.3. Entrenamiento de la red neuronal

La red neuronal se entrena mediante aprendizaje supervisado usando los conjuntos de datos extraídos de los registros de peticiones. Se ha divido los registros de datos en tres conjuntos: entrenamiento, validación y test, conteniendo cada uno un 75 %, 15 % y 15 % de las peticiones totales, respectivamente. El primero de ellos se utiliza para entrenar a la red neuronal. El segundo se utiliza para validar que la red neuronal está generalizando correctamente y no se produce sobre-entrenamiento (*Overfitting*). El último conjunto de datos se utiliza para evaluar el rendimiento de la red neuronal, sin afectar al proceso de aprendizaje de la misma.

Cada registro de entrenamiento consiste en un vector de entrada con los parámetros normalizados de actualidad, frecuencia y tamaño, y la salida *booleana* que es conocida. Los pesos se ajustan utilizando el algoritmo *backpropagation*, que emplea el método de gradiente descendente para minimizar el error cuadrático medio (MSE - *Minimum Square Error*) entre la salida de la red neuronal y las salidas esperadas para cada conjunto de entradas (Mitchell, 1997). La red se entrena en *batch mode*, de forma que los pesos y *biases* se actualizan únicamente después de que se presentan todos los registros de entrenamiento.

Se ha empleado el algoritmo *pocket* (Gallant, 1990), el cual guarda los mejores pesos encontrados durante la fase de validación.

Las simulaciones se han llevado a cabo utilizando los registros de acceso de los servicio CartoCiudad, IDEE-Base y PNOA, descritos en el Capítulo 5. Estos registros han sido previamente filtrados para usar sólo peticiones válidas, de forma que la red neuronal se entrena únicamente con peticiones cacheables. Cada registro de datos contiene las peticiones recibidas por cada servicio del 1 al 7 de Marzo de 2010. Entre estas fechas se han recibido un total de 25.922, 94.520, y 186.672 peticiones de mapa válidas en los servicios CartoCiudad, IDEE-Base y PNOA, respectivamente.

Para la realización de las simulaciones se ha hecho uso del *Toolbox* de redes neuronales de Matlab (Demuth et al., 1992).

6.1.4. Resultados

Como medida del rendimiento conseguido con la red neuronal se ha utilizado el CCR (*Correct Classification Ratio*), que calcula el porcentaje de las peticiones que han sido clasificadas correctamente frente al número total de peticiones procesadas.

Tabla 6.2: CCR (%) obtenidos durante las fases de entrenamiento, y validación y test para los distintos servicios

	CartoCiudad	PNOA	IDEE-Base
entrenamiento	76.5952	96.5355	75.6529
validación	70.2000	97.1985	77.5333
test	72.7422	97.4026	82.7867

En las Figuras 6.3 - 6.5 se muestran los CCRs obtenidos durante las fases de entrenamiento, validación y test para los servicios Cartociudad, IDEE-Base y PNOA. Como se puede observar, la red neuronal es capaz de clasificar correctamente la *cacheabilidad* de las peticiones, alcanzando valores de CCR, con el conjunto de datos de test, comprendidos

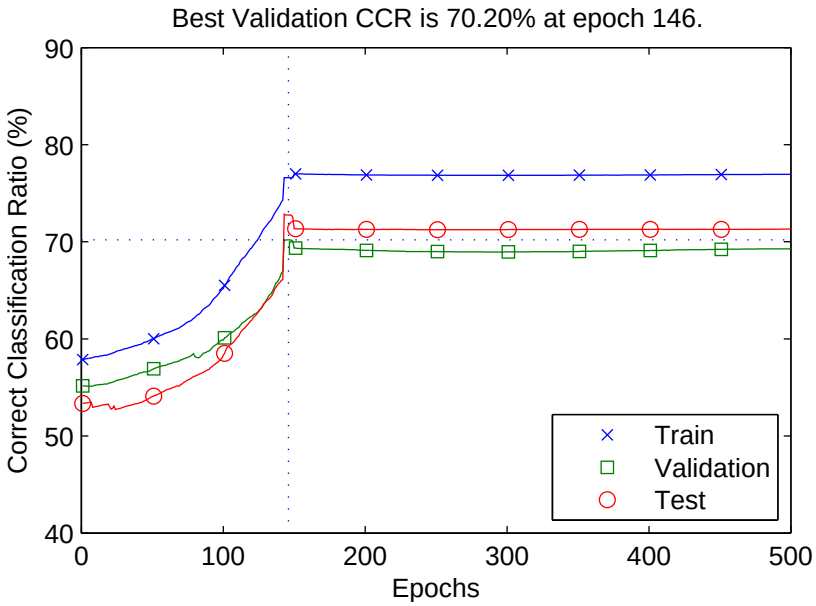


Figura 6.3: CCR obtenidos para Cartociudad.

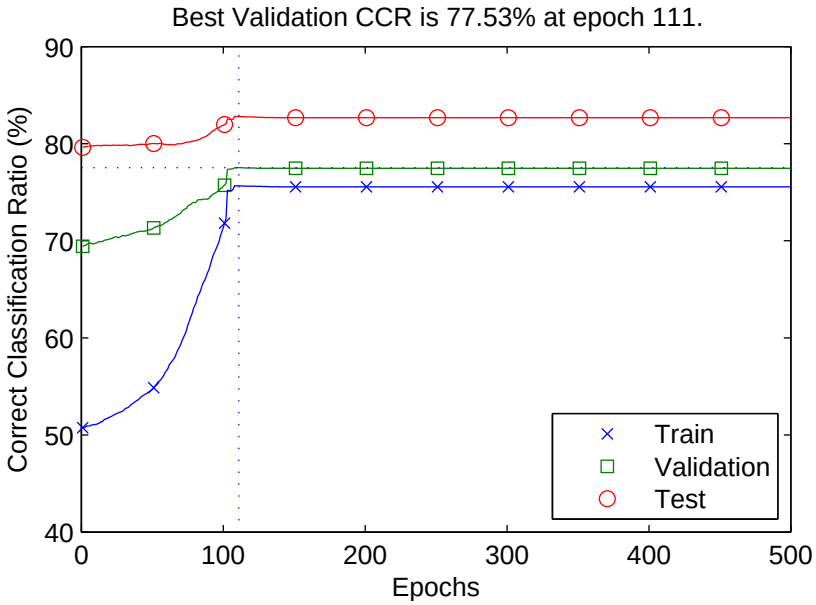


Figura 6.4: CCR obtenidos para IDEE-Base.

entre el 72 % y el 97 %, como se recoge en la Tabla 6.2. Considérese, con fines comparativos, que los valores de CCR obtenidos en (Cobb y ElAarag, 2008) estaban comprendidos entre el 85 % y el 88 %.

Como se puede observar, la red se estabiliza a un valor aceptable de CCR entre 100 a 500 iteraciones, de forma comparable a (Romano y ElAarag, 2011).

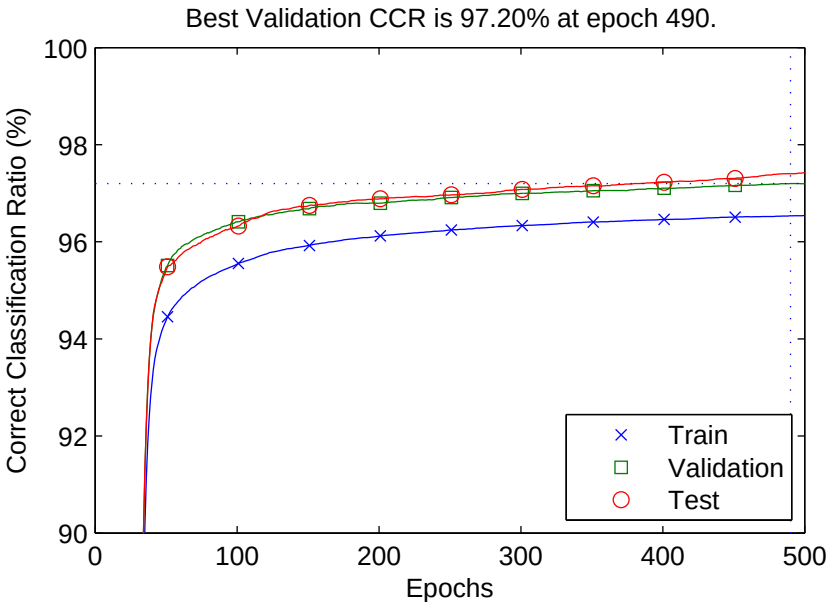


Figura 6.5: CCR obtenidos para PNOA.

6.2. Estrategia de reemplazo de caché Spatial-LFU

Actualmente existe un amplio abanico de estrategias de reemplazo que son comúnmente utilizadas en el ámbito de las cachés Web. Entre estas estrategias podemos destacar las populares LRU (*Least Recently Used*), LFU (*Least Frequently Used*) ó Size (Podlipnig y Böszörmenyi, 2003), por citar algunas. Sin embargo, ninguna de estas estrategias tiene en cuenta la localidad espacial de los objetos gestionados por una caché de mapas. En este trabajo se propone la definición de una variante de la estrategia LFU, bautizada como «Spatial-LFU», que aprovecha la autocorrelación espacial de las peticiones para simplificar el algoritmo y así disminuir la sobrecarga que supone el mantener contadores individuales para todos los objetos. Para evaluar el rendimiento de este nuevo algoritmo se han realizado simulaciones utilizando registros reales de peticiones extraídos de los *logs* del servicio WMS-C de Cartociudad. Los resultados de los experimentos demuestran que pueden mantenerse grandes tasas de acierto de caché, reduciendo significativamente la sobrecarga derivada del uso de este algoritmo.

Como ya se ha comentado, los servicios de mapas teselados consiguen una aceleración del servicio al servir imágenes de mapa pregeneradas mediante el uso de cachés, generalmente situadas en el lado del servidor. Sin embargo, cuando la cartografía a servir cubre una región geográfica extensa para un elevado número de niveles de resolución, los requisitos de almacenamiento resultan a menudo prohibitivos, por lo que generalmente se recurre al uso de cachés parciales limitadas en tamaño. En este escenario, cuando la caché está llena, una política de reemplazo debe decidir qué objetos serán borrados para dejar hueco a otros nuevos.

Sin embargo, la mayor parte de gestores de caché de mapas analizados, como TileCache o MapProxy, no implementan ninguna estrategia de reemplazo. Otras, como GeoWebCache, tan solo implementan estrategias muy básicas, como LRU y LFU, que no tienen en cuenta la componente espacial de los objetos:

- LRU (*Least Recently Used*): elimina de la caché el objeto menos referenciado recientemente.
- LFU (*Least Frequently Used*): selecciona para el reemplazo al objeto que ha sido pedido con menor frecuencia. Existen dos variedades de este algoritmo:
  - InCache-LFU: Tan sólo mantiene los contadores de los objetos que se encuentran almacenados en la caché. Cuando un objeto se elimina de la caché, también se pierden las estadísticas del mismo. La principal ventaja de esta variante es que se incurre en una menor sobrecarga.
  - Perfect-LFU: Considera todas las peticiones realizadas a cada objeto. Los contadores se mantienen entre reemplazos. De esta forma se garantiza que los contadores representan todo el histórico anterior, a costa de incurrir en una mayor sobrecarga de almacenamiento para recoger las estadísticas de todos los objetos.

6.2.1. Comparativa de rendimiento entre las estrategias de reemplazo más populares

Con el objetivo de evaluar el rendimiento de las estrategias anteriores, así como de otros algoritmos, se ha implementado un prototipo de caché (ver Apéndice A.4). Este prototipo simula el siguiente modo de funcionamiento, típico en la mayoría de implementaciones prácticas existentes (Podlipnig y Böszörmenyi, 2003): cuando se produce un fallo de caché, el objeto solicitado se almacena en la caché. Cuando el tamaño de ésta supera un límite pre-establecido, se elimina un cierto número de objetos de la caché para dejar espacio libre a nuevos objetos. La caché utiliza dos marcas HM (*high mark*) y LM (*low mark*), con $HM > LM$. Cuando el tamaño de la caché excede HM, una política de reemplazo elimina objetos hasta que se alcanza el nivel LM.

Sobre este prototipo se han realizado simulaciones utilizando los registros de peticiones disponibles del servicio WMS-C de Cartociudad. Se ha utilizado un total de 200.000 peticiones correspondientes a la capa "Callejero".

En la Figura 6.6 se muestran los resultados de las simulaciones para los algoritmos InCache-LFU, Perfect-LFU, LRU, y el algoritmo óptimo de Belady OPT, que selecciona para el reemplazo el objeto que vaya a ser referenciado más tarde. Este último, aunque se trata de un algoritmo ideal y no es realizable en la práctica, es útil para determinar cuál es el máximo rendimiento que puede conseguirse, con fines comparativos.

Se observa cómo los algoritmos que explotan el parámetro frecuencial de las peticiones (InCache-LFU y Perfect-LFU) obtienen mejores resultados que el algoritmo LRU, el cual utiliza la localidad temporal para la toma de decisiones.

Se observa también que, de entre las dos variantes de LFU analizadas, Perfect-LFU consigue mayores tasas de acierto de caché. Esto se debe a que, como se comentó anteriormente, este algoritmo tiene en cuenta todo el histórico de peticiones.

6.2.2. Reducción de la sobrecarga de los algoritmos LFU

En el apartado anterior se ha mostrado que los algoritmos LFU ofrecen un buen rendimiento en su aplicación a una caché de mapas. En este trabajo se pretende reducir la sobrecarga derivada del uso de estos algoritmos. Para ello se explota el principio de localidad espacial entre las peticiones. Esta dependencia espacial fue cuantificada mediante

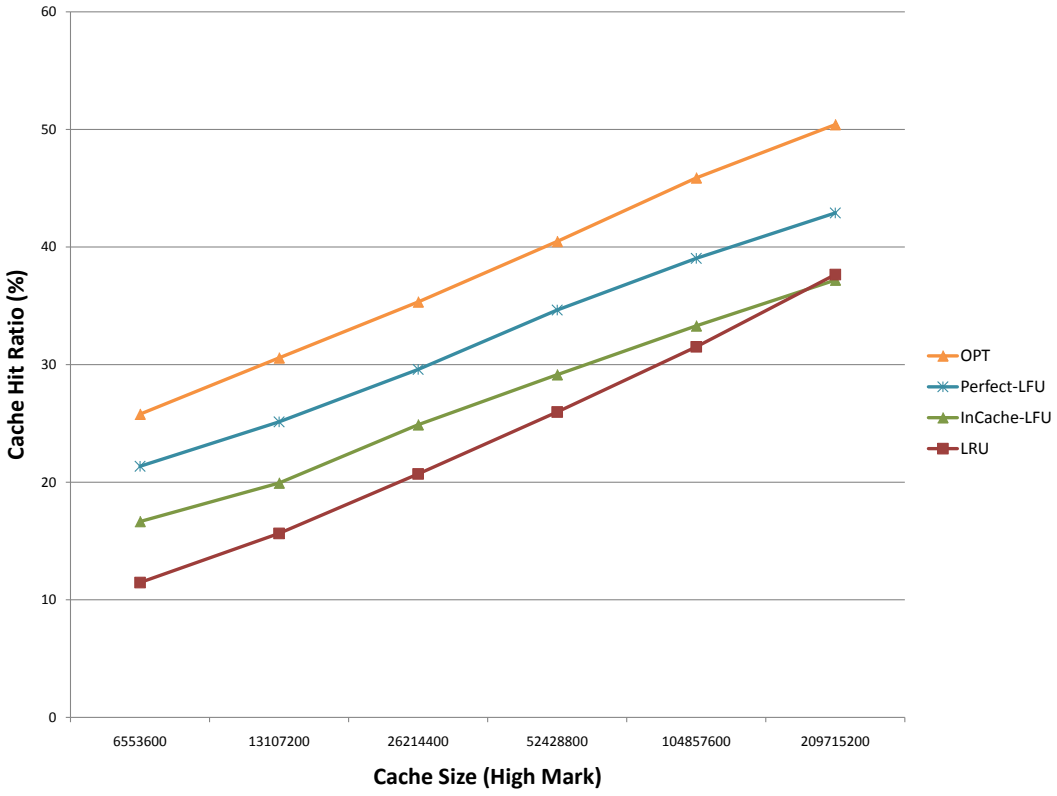


Figura 6.6: Comparativa del porcentaje de aciertos de caché usando los algoritmos InCache-LFU, Perfect-LFU, LRU y el algoritmo óptimo de Belady OPT.

los índices de Moran y Geary en el Capítulo 5.10. Con el sistema propuesto, en lugar de almacenar los contadores de teselas individuales, se agrupan teselas vecinas para compartir un mismo contador.

La simplificación se realiza según se describe en el siguiente pseudo-código:

```
function [x',y',z] = getSimplifiedTileIndex(tile,delta,refRes)

    x = tile.x;
    y = tile.y;
    z = tile.res;

    if(z>refRes)
        if(z-refRes<delta)
          delta = z - refRes;
        end
        x' = floor(x./(2^delta));
        y' = floor(y./(2^delta));
    else
        x' = x;
        y' = y;
```

```
    end

end
```

donde las peticiones de la tesela $T(x, y, z)$ incrementan el contador del grupo de teselas $T_g(x', y', z)$. De esta forma se reduce el espacio muestral de posibles contadores en función del valor de delta (Δ).

Para $\Delta = 0$ no se simplifica nada. Con $\Delta = 1$ se simplifica al espacio de teselas definido por el nivel anterior en la pirámide de escalas. De forma general, el espacio de teselas definido en el nivel n de la pirámide, se reduce al definido en el nivel $n - \Delta$, con lo que el máximo ahorro en número de contadores que puede conseguirse es de un factor de 4^{Δ}.

Dado que el número de teselas aumenta generalmente de forma exponencial con la escala de representación, son los niveles inferiores de la pirámide los que requieren un tratamiento especial. Por ello, se selecciona un nivel de resolución de referencia $refRes$ por encima del cuál no se lleva a cabo ninguna simplificación. En nuestro estudio se ha utilizado el nivel 12 como referencia, que engloba 400×256 teselas en el área seleccionada correspondiente a la Península Ibérica.

Para cada grupo de teselas T_g se mantiene un contador h (*hits*) de peticiones que se incrementa con cada petición de las teselas que agrupa, y otro contador (n) que recoge el número de teselas de ese grupo que se encuentran en caché.

A partir de estos dos contadores, la métrica de este nuevo algoritmo se define como el cociente de estos contadores: $metrica = h/n$

Se ha realizado esta normalización para responder al comportamiento esperado:

- Si $h \downarrow$ y $n \uparrow \Rightarrow metrica \downarrow$: se han recibido pocas peticiones y además están repartidas entre muchos objetos, por lo que cada objeto recibe de media un bajo número de peticiones. La métrica es baja, por lo que los objetos de este grupo son buenos candidatos para el reemplazo.
- Si $h \uparrow$ y $n \downarrow \Rightarrow metrica \uparrow$: se han recibido muchas peticiones y además están repartidas entre pocos objetos, por lo que cada objeto recibe de media un elevado número de peticiones. La métrica es alta, por lo que los objetos de este grupo se mantienen en caché.

Esta simplificación de los algoritmos LFU en base a la localidad espacial de las peticiones da lugar a una nueva familia de algoritmos, que hemos denominado como Spatial-LFU, y que al igual que su homóloga, tiene dos variantes:

- InCache-Spatial-LFU-{Δ}: cuando se elimina de la caché la última tesela ($n = 0$) de un grupo de teselas T_g, se eliminan los contadores de este grupo.
- Perfect-Spatial-LFU-{Δ}: los contadores de grupo se persisten entre reemplazos, aunque se eliminen todas las teselas del grupo.

donde Δ es la delta de simplificación.

6.2.3. Resultados

A continuación se muestran los resultados de las simulaciones realizadas por el sistema propuesto utilizando los nuevos algoritmos Spatial-LFU implementados.

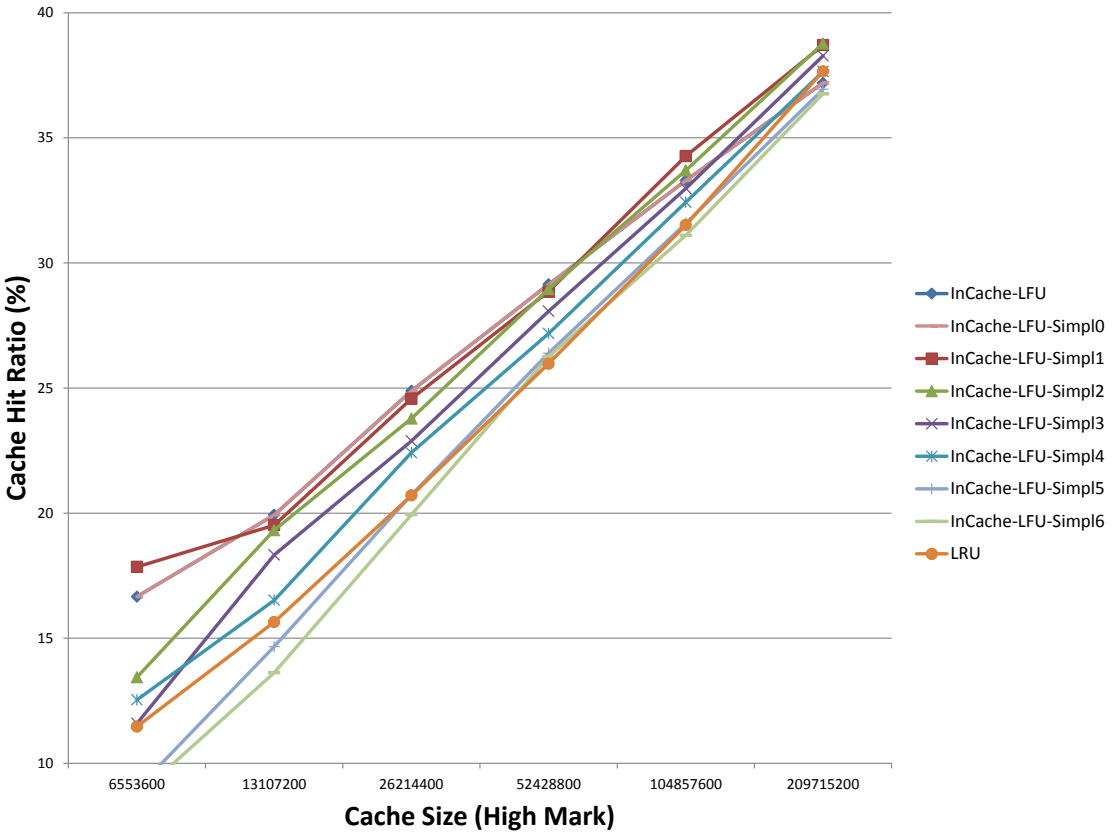


Figura 6.7: Comparativa del porcentaje de aciertos de caché usando los algoritmos InCache-LFU Simplificados frente a InCache-LFU y LRU.

6.2.3.1. InCache-Spatial-LFU

En la Figura 6.7 se recogen las tasas de acierto de caché conseguidas utilizando los nuevos algoritmos `InCache-SpatialLFU` con deltas de simplificación entre 0 y 6, para distintos tamaños de caché. Evidentemente, con un factor de simplificación $\Delta = 0$ (no se simplifica nada) los resultados coinciden con `InCache-LFU`. Para algunos tamaños de caché incluso se consigue una mejora de rendimiento, para $\Delta = 1$ y $\Delta = 2$. Se observa que, a pesar de que la carga (el número de teselas que abarca cada grupo) disminuye en un factor de 4^{Δ}, las tasas de acierto disminuyen de forma lineal respecto a Δ.

En las Figuras 6.8 y 6.9 se muestra la evolución en el consumo de recursos (el número de registros con estadísticas) para diversos tamaños de caché y factores de simplificación. Asimismo, se muestran los tamaños medio y máximo del registro. Considérese, por ejemplo, que para un factor de simplificación $\Delta = 5$ y tamaño de la caché $H = 209715200$ bytes (ver Figura 6.8), el número máximo de estadísticas almacenadas se reduce de 8528 a tan solo 2644 (reducción del 69 %), mientras que la tasa de acierto de caché tan sólo se reduce un 3.899 %.

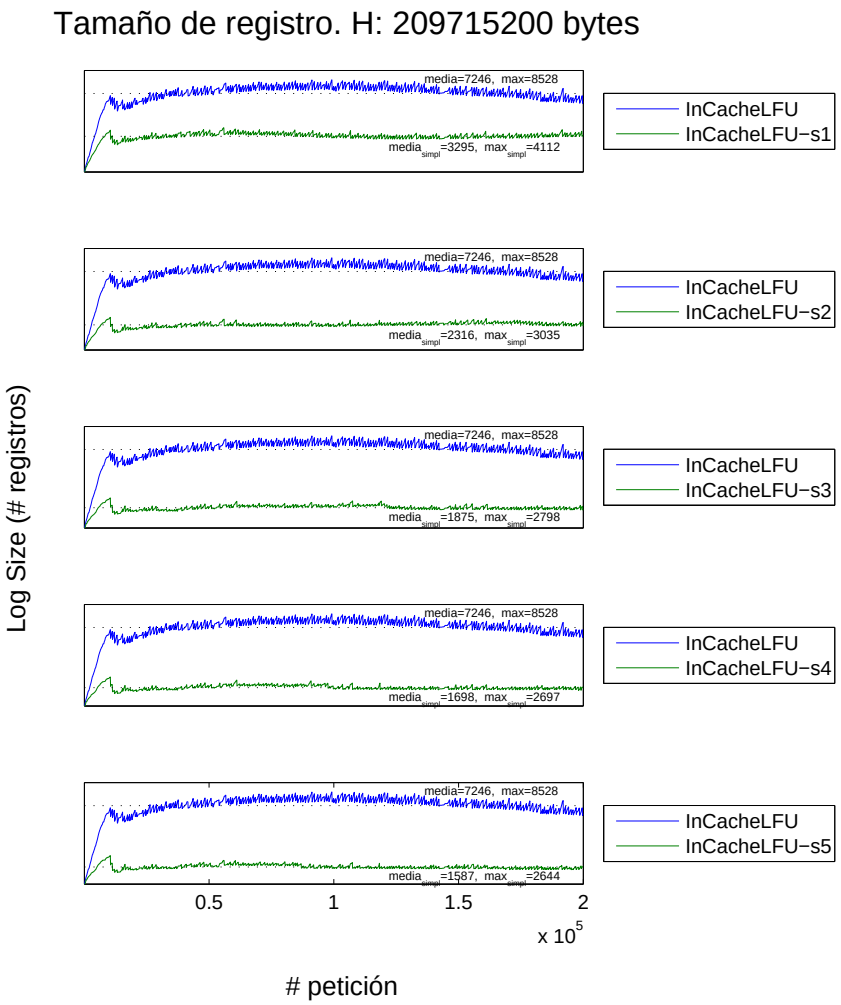


Figura 6.8: Consumo de recursos para estadísticas con el algoritmo InCache-LFU Simplificado, para distintos factores de simplificación. Tamaño de la caché: 209715200 bytes.

6.2.3.2. Perfect-Spatial-LFU

En la Figura 6.10 se recogen las tasas de acierto de caché conseguidas utilizando los nuevos algoritmos `Perfect-SpatialLFU` con deltas de simplificación entre 0 y 6. Evidentemente, con un factor de simplificación $\Delta = 0$ (no se simplifica nada) los resultados coinciden con `Perfect-LFU`. La tasa de acierto se mantiene por encima de la conseguida para `InCache-LFU`, a pesar de que disminuye a medida que la simplificación aumenta.

En la Figura 6.11 se muestra una comparativa entre el consumo de recursos usando los algoritmos Perfect-LFU Simplificados y Perfect-LFU. Se observa que al simplificar con un factor $\Delta = 1$ el número de registros se reduce de 92061 a tan sólo 40874 (reducción del 55.6 %), mientras que la tasa de acierto no se ve apenas afectada, incluso se consiguen mayores tasas de acierto para algunos tamaños de caché. Esta mejora se produce debido a que se aprovecha indirectamente el efecto de localidad espacial entre las peticiones.

Los resultados obtenidos mediante las simulaciones realizadas demuestran que la estrate-

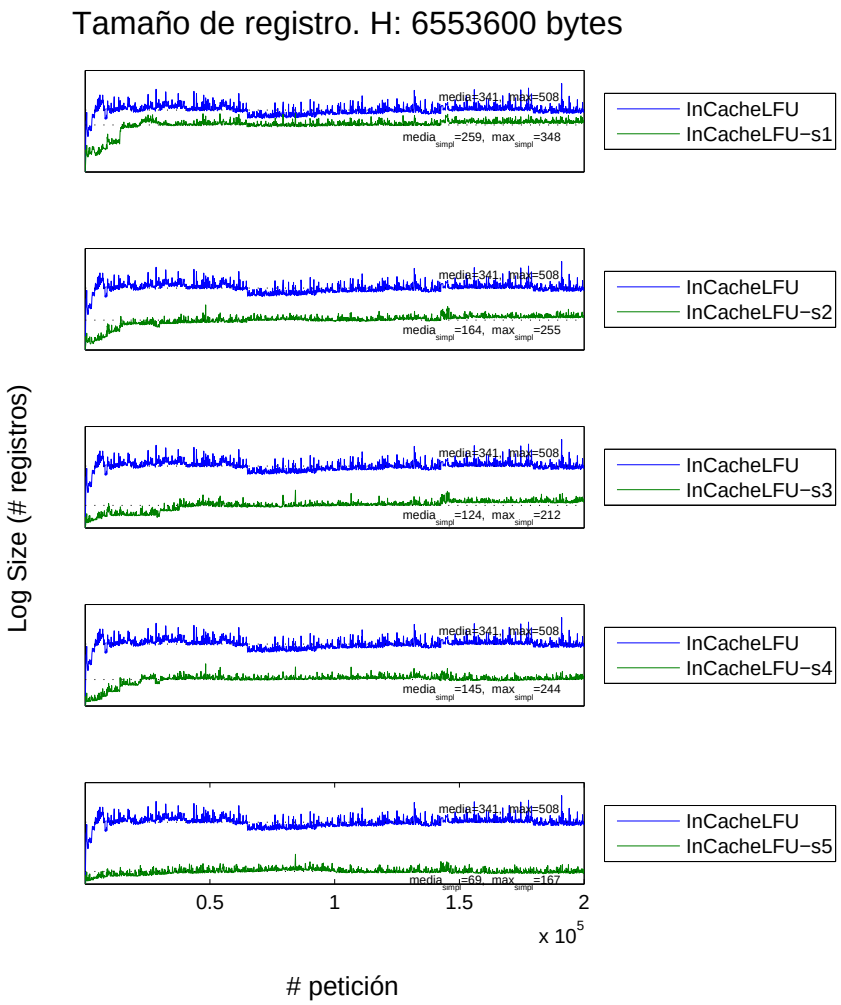


Figura 6.9: Consumo de recursos para estadísticas con el algoritmo InCache-LFU Simplificado, para distintos factores de simplificación. Tamaño de la caché: 6553600 bytes.

gia Perfect-LFU ofrece un mejor rendimiento que la estrategia InCache-LFU, a costa de una mayor sobrecarga en el consumo de recursos. Las variantes Perfect-Spatial-LFU e InCache-Spatial-LFU propuestas permiten reducir la sobrecarga derivada del almacenamiento de los contadores respecto a sus estrategias homólogas, con una reducida penalización de rendimiento.

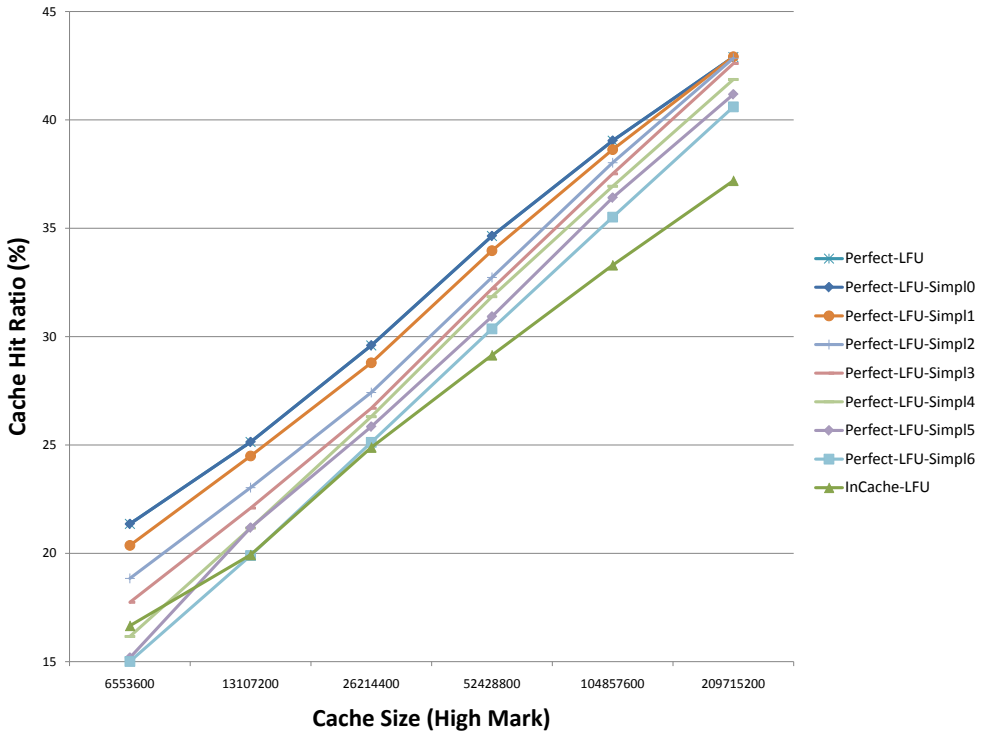


Figura 6.10: Comparativa del porcentaje de aciertos de caché usando los algoritmos Perfect-Spatial-LFU frente a Perfect-LFU e InCache-LFU.

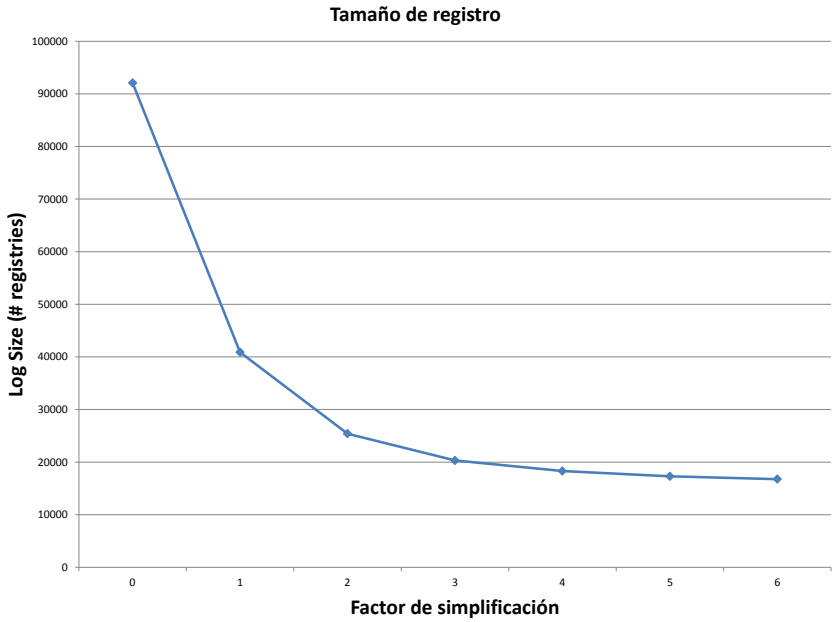


Figura 6.11: Comparativa del consumo de recursos usando los algoritmos Perfect-Spatial-LFU frente a Perfect-LFU.

Capítulo 7

Políticas de precarga de caché

Resumen: Durante la puesta en marcha de un servicio de mapas la caché se encuentra inicialmente vacía, por lo que los usuarios experimentan una baja QoS. Ésta aumenta a medida que la caché se va poblando con teselas, lo cual se puede hacer dejando que las teselas se vayan cacheando bajo demanda la primera vez que son pedidas por los usuarios. La otra posibilidad, conocida como precarga o *seeding*, consiste en precargar las teselas de forma automática anticipándose a las peticiones de los usuarios, con lo que se elimina la penalización inicial existente en el cacheo bajo demanda. En este capítulo se propone, por una parte, un mecanismo de precarga inicial basado en un modelo descriptivo que utiliza el histórico del servicio. Por otra parte, se proponen sendos modelos predictivos, uno basado en regresiones lineales y otro basado en redes neuronales, que se aprovechan de la presencia de elementos directrices de las peticiones de los usuarios.

En el Capítulo 5 se recogieron múltiples trabajos para la precarga de teselas utilizando el patrón global de accesos a largo plazo de todos los usuarios. Sin embargo, la mayor parte de los trabajos encontrados en la literatura obvian un factor muy importante, la información geográfica subyacente. Los usuarios no visitan las teselas de mapa de forma aleatoria, sino que existen ciertas regiones de mapa más populares que otras. En cualquier caso, estos estudios no consideran qué motiva a los usuarios a visitar estas regiones.

Quinn y Gahegan (2010) proponen un modelo para determinar regiones populares de mapa para la precarga de teselas considerando fenómenos geográficos. Este modelo tiene en cuenta aquellas variables que, en opinión de los investigadores, recaban un mayor interés por parte de los usuarios según las observaciones de la herramienta *Hotmap*, tales como como zonas costeras, principales vías de transporte y zonas altamente pobladas. Sin embargo, sus autores no disponen de ningún mecanismo para la identificación automática de estas regiones de interés. Por otra parte, el modelo que proponen no tiene en cuenta que algunas de estas *features* pueden ser mucho más directoras de las peticiones de los usuarios que otras, sino que todas contribuyen por igual a la definición de las áreas de interés. Por lo tanto, todas las teselas que pertenecen a cualquiera de estas áreas se seleccionan para ser cacheadas.

A lo largo del documento se habla indistintamente de fenómenos como de *features*. Adoptamos el mismo concepto de fenómeno o *feature* geográfica que el descrito por el *OGC Abstract Specification* (Kottman y Reed, 2009):

> *"A feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth."*

El término «fenómeno» es la traducción adoptada para *feature* según el Glosario Multi-Lingüe[1] publicado por el comité ISO/TC 211 con la terminología de la familia de normas ISO 19100 sobre información geográfica. Por ejemplo, una superposición de temperaturas sobre un mapa meteorológico o un segmento de carretera son ejemplos de fenómenos geográficos.

Evidentemente, una *feature* no contribuye a las peticiones de los usuarios de igual manera en todos los servicios. Por ejemplo, en una IDE sobre los transportes, como la que posee la Dirección General de Tráfico (DGT)[2] es de esperar que los elementos definidos por los viales sean más directores de las peticiones de los usuarios que, por ejemplo, las *features* definidas por las zonas costeras. Por ello, aquí se propone un modelo que permita considerar de forma ponderada las contribuciones de múltiples *features*.

A continuación en este capítulo se propone:

- Una arquitectura genérica para la precarga de teselas utilizando un catálogo general de fenómenos geográficos y la historia del servicio.
- Un mecanismo de precarga basado en un modelo descriptivo que utiliza el histórico de peticiones del servicio.
- Un mecanismo de precarga basado en un modelo predictivo que asigna, de forma automática, prioridades a las teselas en base a una combinación lineal de variables geográficas en un catálogo general.
- Un método para la parametrización automática de este modelo mediante un estimador *Ordinary Least Squares* usado para ajustar el modelo al registro observado de accesos pasados.
- Un sistema inteligente basado en el uso de redes neuronales para asignar automáticamente prioridades a las teselas a partir de un conjunto de fenómenos geográficos, con un breve período de entrenamiento a partir de registros de peticiones.

Esta aproximación supera las limitaciones encontradas en (Quinn y Gahegan, 2010). En primer lugar, se evita la necesidad de un administrador para la selección manual de las variables de interés, pues el modelo selecciona automáticamente los pesos adecuados para cada una de las variables predictoras. En segundo lugar, asigna probabilidades no-binarias a las teselas, lo cuál permite establecer prioridades a ciertas regiones sobre otras. Por último, es adaptativo; Puede ejecutarse periódicamente de forma que si las necesidades e intereses de los usuarios varían con el tiempo, el modelo puede ajustarse según corresponda.

Aparte de las implementaciones anteriores, mediante regresiones lineales y redes neuronales, se ha llevado a cabo una particularización del modelo propuesto mediante algoritmos genéticos. Para ello, se ha aprovechado el conocimiento de esta técnica adquirido durante su aplicación en el campo de *e-learning* (Verdú et al., 2012a,b). Sin embargo, con esta implementación se obtienen medidas de rendimiento de caché muy por debajo de con el resto de implementaciones exploradas, por lo que esta línea ha sido abandonada.

[1] http://www.isotc211.org/Terminology.htm
[2] http://www.dgt.es

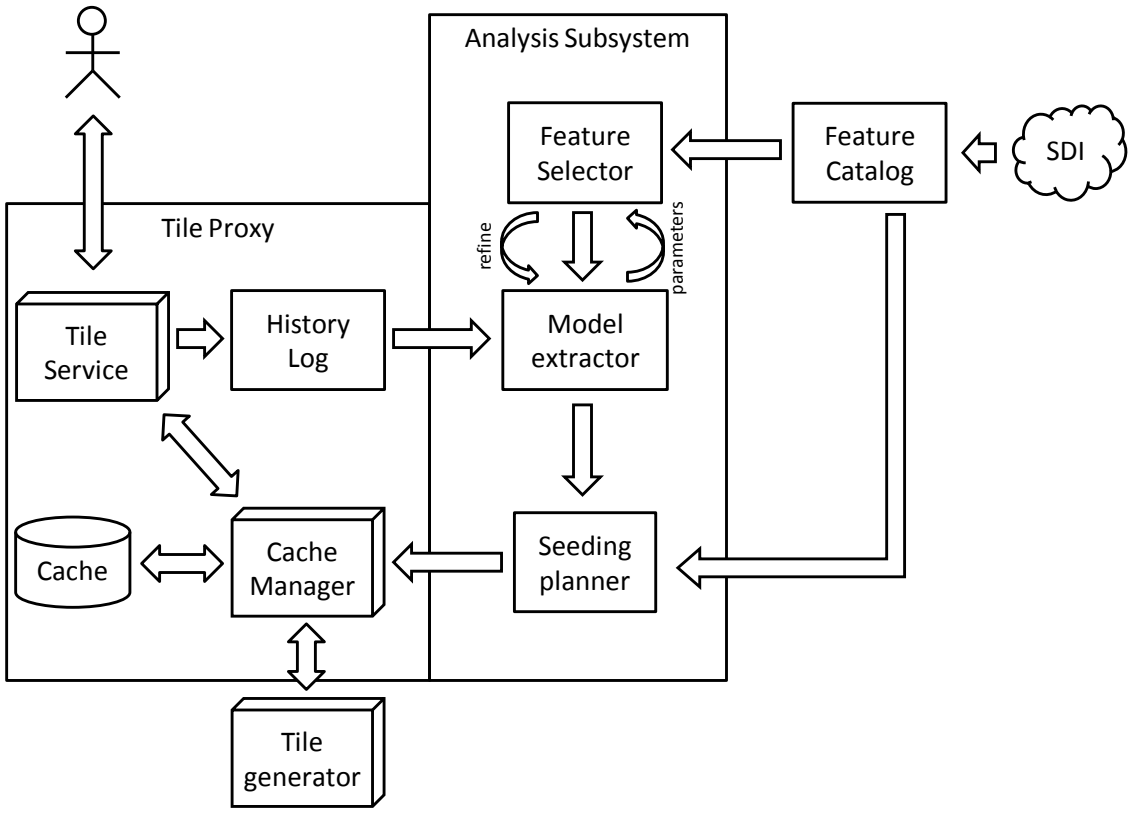


Figura 7.1: Arquitectura propuesta para la precarga de teselas a partir de un catálogo general de fenómenos geográficos y la historia del servicio.

7.1. Arquitectura del Sistema

A continuación se describe la arquitectura propuesta para la precarga de teselas. Esta arquitectura trata de anticipar posibles peticiones futuras a partir de un catálogo de fenómenos no-restringido y la historia previa del servicio (ver Figura 7.1).

Dicha arquitectura consta de dos módulos principales: *proxy* de teselas y subsistema de análisis. El *proxy* de teselas contiene la lógica presente en la mayoría de implementaciones existentes de caché de teselas, como GeoWebCache (Liu y Nie, 2010) o WMSCWrapper. El subsistema de análisis se concibe como un *plug-in* que incorpora funcionalidad inteligente para la precarga de objetos a un *proxy* de teselas existente. Este subsistema de análisis emplea un conjunto de fenómenos geográficos como variables predictoras para la estimación de peticiones futuras a partir de la historia pasada del servicio. Finalmente, estas predicciones se utilizan para precargar aquellas teselas que se espera que sean más pedidas.

7.1.1. Proxy de teselas

El servicio de teselas (*TileService*) ofrece interfaces estándar para atender las peticiones de mapa de los usuarios, tales como WMTS y WMS-C, u otros esquemas propietarios de teselado como los utilizados por Google Maps o Microsoft Bing Maps. Al recibir una

petición de mapa, el *TileService* consulta al gestor de caché (*CacheManager*) para averiguar si la tesela solicitada está almacenada en la caché. En caso afirmativo, el objeto se devuelve directamente a partir de la misma. En caso contrario, el gestor de caché solicita al generador de teselas (*TileGenerator*) que obtenga la tesela solicitada, ya sea a partir de un servicio WMS o cualquier otra fuente de datos. Entonces, la tesela obtenida se introduce en la caché y se devuelve al cliente a través del *TileService*.

7.1.2. Gestor de caché

El gestor de caché mantiene un registro o *log* de las peticiones recibidas, para su uso en el subsistema de análisis. Almacenar las estadísticas de teselas individuales pertenecientes a las escalas de representación de mayor detalle podría suponer una sobrecarga significativa. Por tanto, el espacio de teselado se ha simplificado a la rejilla definida por un nivel superior, según el procedimiento expuesto en el Capítulo 3.

7.1.3. Catálogo de fenómenos geográficos

Puede obtenerse un catálogo general de fenómenos geográficos a partir de una IDE (Infraestructura de Datos Espacial) mediante servicios estándar de OGC, como los servicios *Web Feature Service* (WFS) (Vretanos, 2005) o *Web Coverage Service* (WCS) (Whiteside y Evans, 2008), devueltos en formato vectorial y ráster, respectivamente. Además, puede emplearse el servicio CSW (*Catalog Service for the Web*) (Nebert et al., 2007) de OGC para descubrir posibles recursos de interés a partir de metadatos.

Puede resultar de utilidad poblar el catálogo con los mismos fenómenos geográficos utilizados para el renderizado del mapa (ver Figura 2.1, pág. 10). Dado que estas *features* serán visualizadas por los usuarios, parece razonable considerarlas como elementos directores de las peticiones de los usuarios. Sin embargo, este argumento puede ser engañoso, ya que los clientes de mapas generalmente realizan superposiciones de múltiples capas incluso procedentes de distintos servicios. Por esta razón, es una buena práctica poblar el catálogo con *features* adicionales mas allá de las utilizadas durante el proceso de generación de la imagen de mapa. Es responsabilidad del sistema inteligente determinar qué fenómenos utilizar de entre los presentes en el catálogo.

Cada fenómeno geográfico incluido en el catálogo se mapea al espacio de teselado definido en el área de estudio. De esta forma, a cada tesela se le asigna un valor extraído a partir de un atributo contenido o asociado a cada *feature*. Considérese, con fines ilustrativos, un fenómeno geográfico de tipo "autopista". La variable asociada a este fenómeno podría contener un valor binario indicando si dicho fenómeno intersecta o no el área cubierta por cada tesela. Podría definirse también como una variable continua consistente en una medida de la distancia entre la tesela muestreada y la autopista más cercana. Asimismo, podría definirse incluso como una medida del número de vehículos que circulan por cada autopista concreta por unidad de tiempo. En cualquier caso, a cada tesela se le asigna un conjunto de valores, uno para cada fenómeno geográfico.

Otro ejemplo de fenómeno, consistente en las estadísticas de acceso a las teselas de un servicio de mapas en un determinado nivel de escala, se usa en el modelo descriptivo presentado en la Sección 7.2.

7.1.4. Selector de fenómenos geográficos

Dado que un catálogo puede contener centenares de capas de fenómenos geográficos, el componente *FeatureSelector* se encarga de reducir esta cantidad seleccionando un subconjunto del catálogo general. Para ello, por una parte se aplica un filtro espacial, seleccionando tan solo aquellos fenómenos que intersectan el área de estudio. Para reducir la multicolinealidad entre las variables, el *FeatureSelector* elimina las variables más inter-correladas entre sí, utilizando técnicas de reducción de datos como *Principal Components Analysis* (PCA). Este proceso de selección de fenómenos no tiene en cuenta las peticiones de los usuarios, sino tan sólo las propiedades de los fenómenos geográficos y sus relaciones.

7.1.5. Extractor del modelo

El componente *ModelExtractor* utiliza un algoritmo de aprendizaje automático (*machine learning*) para inferir un modelo probabilístico que permita estimar las peticiones futuras, basándose para ello en el conjunto de fenómenos seleccionado y la historia de accesos pasados. La salida de este modelo es un conjunto de parámetros específico de cada predictor. Por ejemplo, en un caso particular en el que se use un modelo de combinación lineal, caso que se trata posteriormente en las secciones 7.3.2 y 7.2, los parámetros de salida del modelo son los coeficientes calculados para dicha combinación lineal de fenómenos.

7.1.6. Planificador de la tarea de precarga

El planificador de la tarea de *seeding* (*SeedingPlanner*) usa los parámetros de salida del *ModelExtractor*, para generar un *raster* cuyos valores asignan diferentes probabilidades a las teselas. El *SeedingPlanner* va poblando la caché con las teselas precargadas, de acuerdo a las probabilidades asignadas, por orden descendente de probabilidad. Además, tal y como se muestra en la Figura 7.1, los parámetros de salida se realimientan al *FeatureSelector* para refinar el conjunto de fenómenos seleccionado. De esta forma puede, por ejemplo, descartarse aquellas capas de fenómenos cuya contribución a la salida final del modelo sea despreciable.

7.2. Modelo Descriptivo

> Los contenidos del presente apartado han sido parcialmente publicados en (García et al., 2011d) y (García et al., 2011e).

Los modelos descriptivos utilizan los registros de acceso a los servidores de mapas para determinar cuáles son las regiones de mapa más solicitadas por los usuarios. En esta misma línea, la aplicación Web *Hotmap* de Microsoft[3] utiliza los registros de peticiones del servicio de mapas *Bing Maps* para mostrar sobre un mapa el número de veces que se ha pedido cada tesela. *Hotmap* utiliza un *heatmap* o «mapa de calor» para representar de forma gráfica la actividad de los usuarios (Fisher, 2007a,b, 2009). Sin embargo, no ofrece acceso a los propios datos de las peticiones, por lo que limita el posible análisis a una mera prospección visual, no

[3] http://hotmap.msresearch.us/

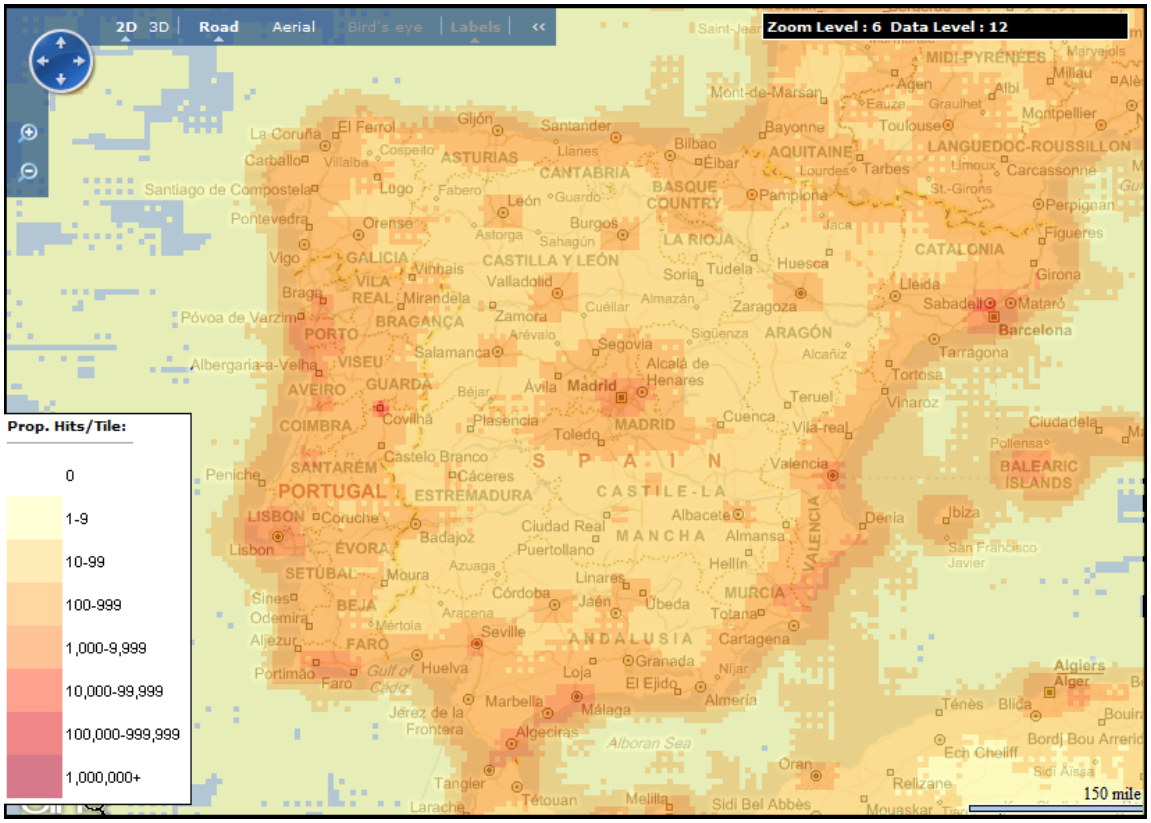


Figura 7.2: Visualización de patrones de acceso en Microsoft Hotmap.

pudiendo hacerse uso de algoritmos automáticos para la extracción de patrones de interés. La Figura 7.2 muestra una captura del uso de esta aplicación.

Para el modelo descriptivo propuesto en esta investigación se han utilizado los registros de peticiones realizadas a los servicios WMS-C de *Cartociudad*, *IDEE-Base* y *PNOA*, descritos en el Capítulo 5.2.

Se ha comprobado que la mayor parte de las peticiones de mapas registradas se realizan sobre la Península Ibérica, como era de esperar dado el ámbito de estos servicios. Por ello, se ha limitado la zona geográfica de estudio a esta región, concretamente al *bounding box* [-11.9971 °W, 5.5371 °E, 32.8711 °S, 46.0107 °N], expresado en coordenadas geográficas EPSG:4326 (ver Figura 7.3). Este *bounding box* está alineado con la rejilla definida por el servicio teselado en el nivel de resolución 12, abarcando 400 y 300 teselas en las direcciones horizontal y vertical, respectivamente, en este nivel.

Dada la naturaleza exponencial de la pirámide de escalas y la imposibilidad de trabajar con las estadísticas de teselas individuales, se ha empleado un modelo simplificado que aproxima la probabilidad de petición de una tesela a partir de las estadísticas recogidas en otro nivel de resolución (l) en la región geográfica cubierta por esa tesela, como se formaliza en la expresión (7.1).

$$\hat{P}_{req,l}\left\{T\left(i,j,n\right)\right\} = \int_{y=y_{n,j}}^{y_{n,j+1}} \int_{x=x_{n.i}}^{x_{n,i+1}} f_{req}\left(x,y,l\right) dxdy \tag{7.1}$$

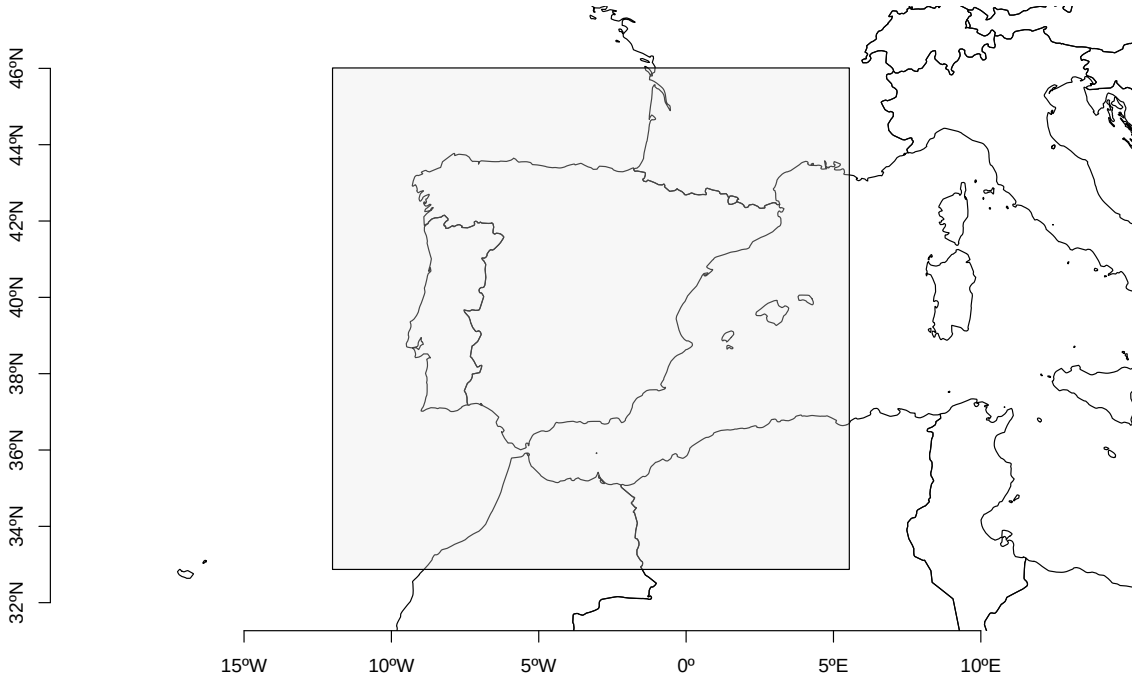


Figura 7.3: Región considerada (área sombreada) para el estudio.

donde

$$f_{req}\left(i,j,n\right) = \frac{n_{h(i,j,n)}}{\Delta x \Delta y N_{total,n}} \tag{7.2}$$

con

$$N_{total,l} = \int_{y=y_{min,l}}^{y_{max,l}} \int_{x=x_{min,l}}^{x_{max,l}} f_{req}\left(x,y,l\right) dxdy \tag{7.3}$$

Concretamente, se ha simplificado el modelo a la rejilla de tamaño 400×300 definida por la región antes mencionada en la escala de resolución 12 ($l = 12$). La estructura piramidal de escalas se transforma, en cierto modo, en una estructura tipo prisma con el mismo número de elementos (400×300) en todas las escalas (ver Figura 7.4).

Dada la naturaleza *booleana* del cacheado (una tesela, o bien se cachea, o por el contrario no lo hace) la salida del modelo viene dada por una función $\Theta_l\left\{T\left(i,j,n\right)\right\}$ de esta misma naturaleza que se aplica sobre la probabilidad de petición de las teselas.

Se define un umbral de probabilidad μ de forma que las teselas precargadas son aquellas cuya estimación de probabilidad de ser pedida supera este umbral:

$$\Theta_l\left\{T\left(i,j,n\right)\right\} = \begin{cases} 1, & \hat{P}_{req,l}\left\{T\left(i,j,n\right)\right\} \geq \mu \\ 0, & en\ otro\ caso \end{cases} \tag{7.4}$$

La elección de este umbral $\mu \in (0,1)$ determinará el número de teselas cacheadas y los aciertos de caché obtenidos. Evidentemente, si $\mu = 0$, se conseguirá un 100 % de aciertos de caché, a costa de cachear todas las teselas. En el lado opuesto, si $\mu = 1$ no se cacheará

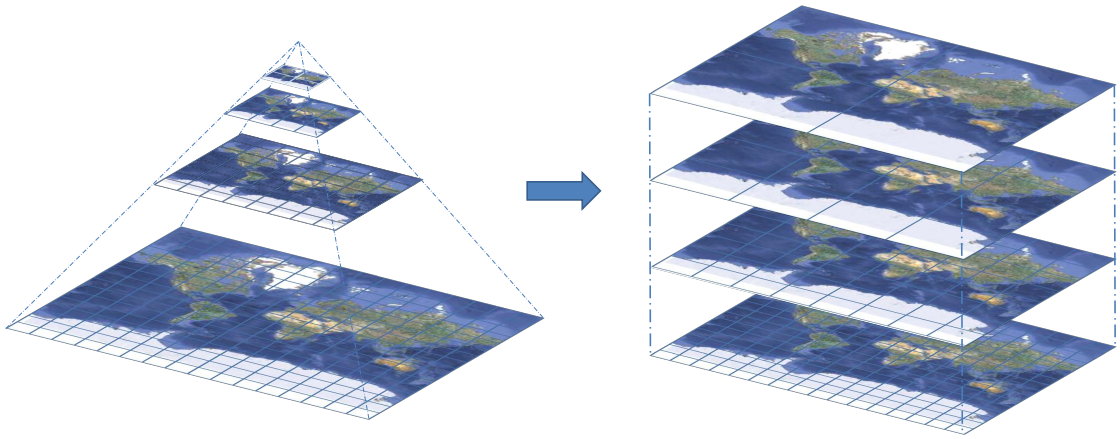


Figura 7.4: Transformación "conceptual" de una pirámide de escalas a un prisma mediante el uso del modelo simplificado.

ningún objeto por lo que no se producirán aciertos de caché. La elección de este umbral debe hacerse de tal forma que se maximice la probabilidad de acierto manteniendo el consumo de recursos por debajo de un nivel dado.

Sea γ una función *booleana* que determina si se ha pedido o no una tesela

$$\gamma\left\{T\left(i,j,n\right)\right\} = \begin{cases} 1, & si\ n_{req}\left\{T\left(i,j,n\right)\right\} > 0 \\ 0, & si\ n_{req}\left\{T\left(i,j,n\right)\right\} = 0 \end{cases} \tag{7.5}$$

$\Psi_l\left(i,j,n\right) \in (0,1)$ representa la fracción de teselas de la celda (i,j,n) del prisma que han sido pedidas alguna vez, para el modelo simplificado a nivel l:

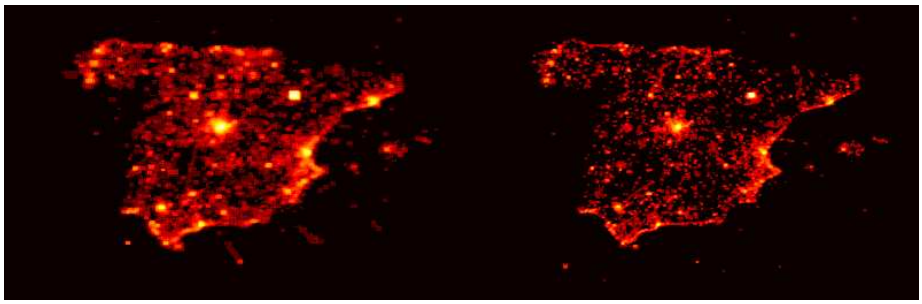
$$\Psi_l\left(i,j,n\right) = \frac{\int_{y=y_{l,j}}^{y_{l,j+1}} \int_{x=x_{l,i}}^{x_{l,i+1}} \gamma\left(x,y,n\right) dxdy}{n_{tot,ijn}} \tag{7.6}$$

Conviene introducir aquí el concepto de "heterogeneidad" de una celda, formalizado según la expresión (7.7). Su interpretación es sencilla; cuando se recoge un elevado número de peticiones en una celda, pero éstas se efectúan sobre un reducido número de teselas pertenecientes a esa celda, se considera que ésta es "heterogénea". Estas zonas requieren un análisis a más bajo nivel.

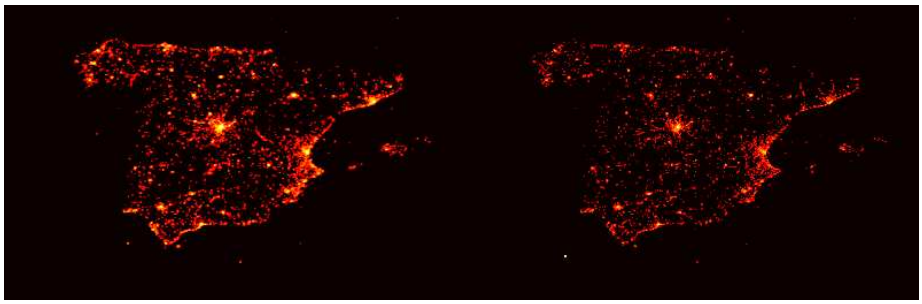
$$heterog_l\left(i,j,n\right) = \hat{P}_{req,l}\left\{T\left(i,j,n\right)\right\}\left(1 - \Psi_l\left(i,j,n\right)\right) \tag{7.7}$$

En las Figuras 7.5, 7.6 y 7.7 se muestran los *heatmaps* o mapas de calor para los registros de peticiones registradas en los servicios de Cartociudad, PNOA e IDEE-Base, respectivamente, según el modelo simplificado al nivel de resolución 12. En estas figuras se observa que existen zonas cuya popularidad es claramente muy superior a la de otras regiones.

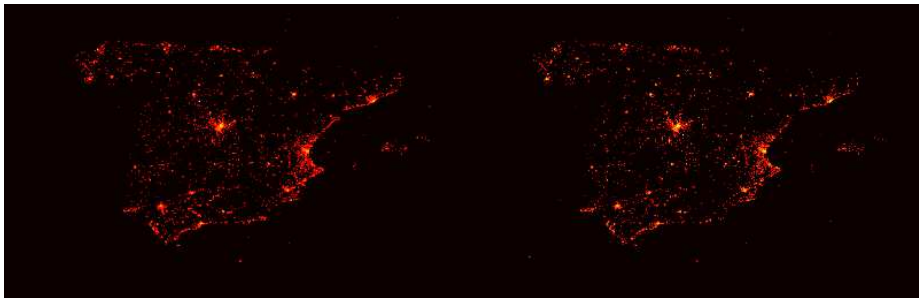
En las Tablas 7.1-7.3 se recogen los porcentajes de acierto para los servicios considerados, utilizando el modelo simplificado a la rejilla del nivel de resolución 12 ($l = 12$), utilizando como umbral la media ($\mu = E\left\{\hat{P}_{req}\right\}$), según (7.4). Estas tablas se interpretan como el porcentaje de aciertos conseguidos en el nivel de resolución identificado por la columna a



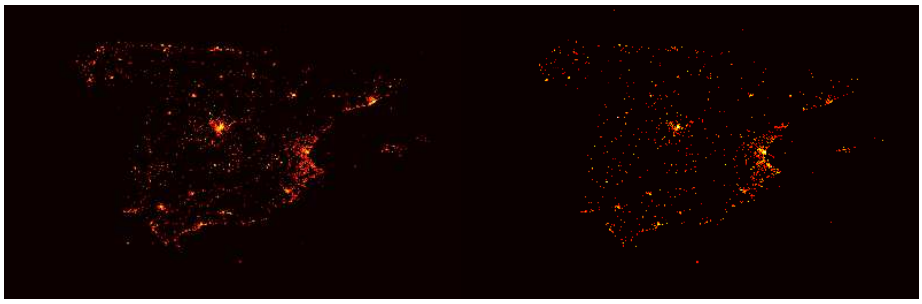
(a) Escala 12 (b) Escala 13



(c) Escala 14 (d) Escala 15

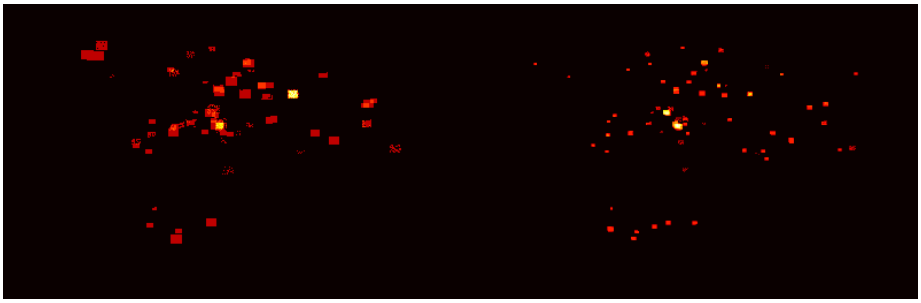


(e) Escala 16 (f) Escala 17

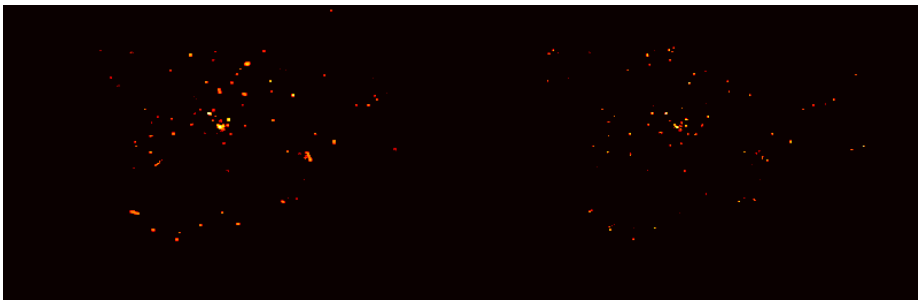


(g) Escala 18 (h) Escala 19

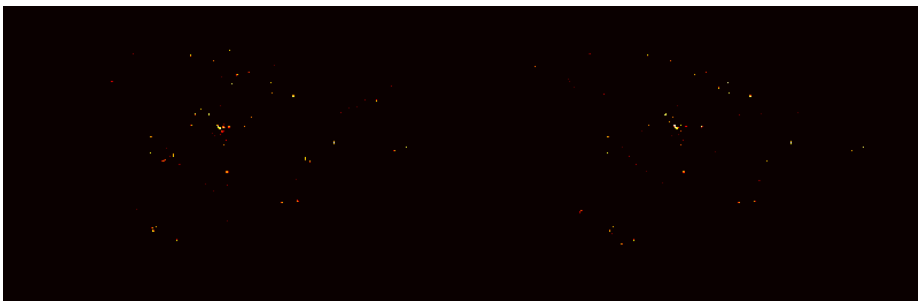
Figura 7.5: Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de Cartociudad propagadas al nivel de resolución 12.



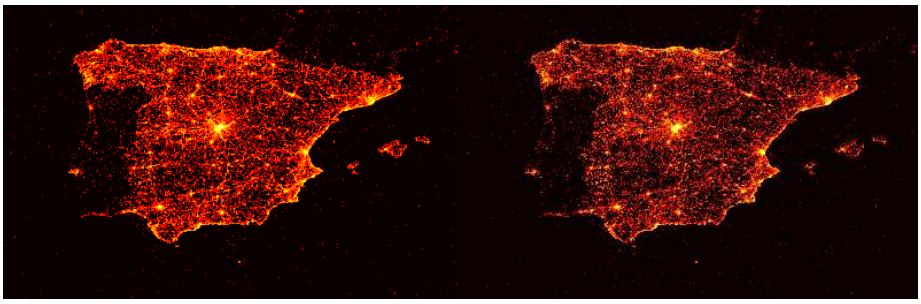
(a) Escala 12 (b) Escala 13



(c) Escala 14 (d) Escala 15

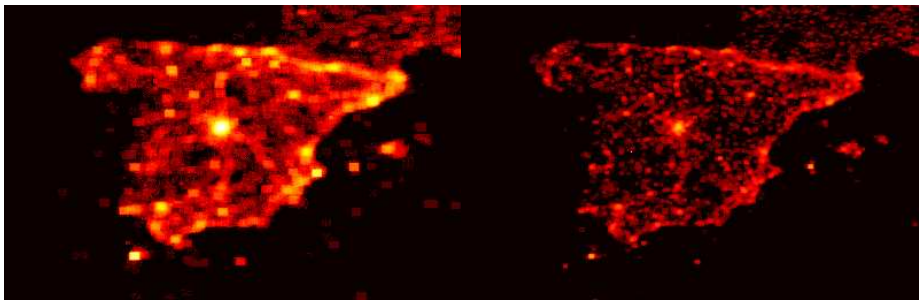


(e) Escala 16 (f) Escala 17

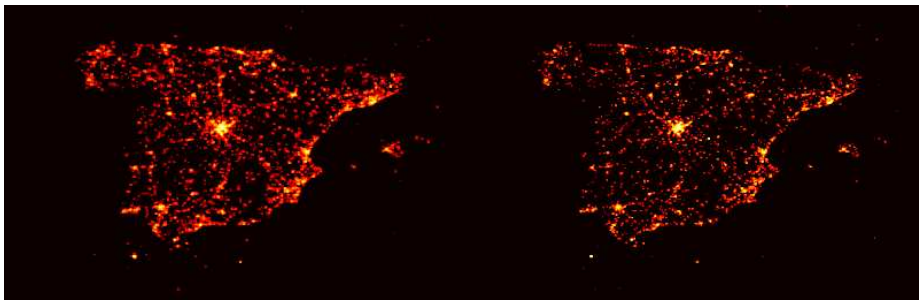


(g) Escala 18 (h) Escala 19

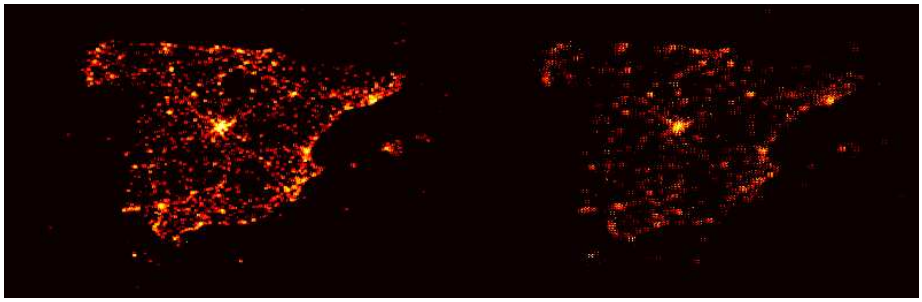
Figura 7.6: Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de PNOA propagadas al nivel de resolución 12.



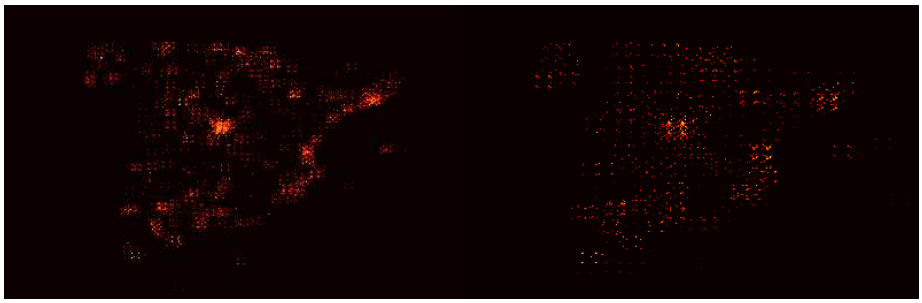
(a) Escala 12 (b) Escala 13



(c) Escala 14 (d) Escala 15



(e) Escala 16 (f) Escala 17



(g) Escala 18 (h) Escala 19

Figura 7.7: Representación gráfica del catálogo de fenómenos generado a partir de los registros de peticiones del servicio WMS-C de IDEE-BASE propagadas al nivel de resolución 12.

Tabla 7.1: Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de Cartociudad, con μ =media.

	12	13	14	15	16	17	18	19	recursos
12	87.72	90.45	91.50	90.27	94.44	95.40	96.07	92.30	18.08
13	69.67	78.86	82.33	83.01	88.67	80.88	91.95	85.09	8.83
14	56.62	69.88	76.49	78.88	84.64	78.87	89.04	80.25	5.40
15	41.40	54.97	65.82	74.59	78.40	74.59	85.51	73.37	3.00
16	33.20	45.61	57.69	66.35	73.81	71.24	84.10	69.05	2.04
17	29.32	42.41	53.36	63.56	54.82	78.21	81.57	66.49	1.61
18	26.39	37.71	50.47	61.57	52.82	67.84	81.79	63.44	1.35
19	15.26	23.44	33.68	44.71	37.99	48.68	72.75	44.52	0.76
prop	67.75	76.35	80.60	84.42	87.49	90.81	92.74	83.80	6.68

Tabla 7.2: Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de IDEE, con μ =media.

	12	13	14	15	16	17	18	19	recursos
12	88.06	85.72	93.76	94.52	96.09	93.75	93.57	79.31	21.02
13	74.86	85.96	90.71	92.18	94.09	89.75	82.49	54.88	15.28
14	48.11	57.73	84.37	86.60	89.16	78.54	59.86	38.77	7.63
15	34.62	45.64	76.00	82.75	81.70	66.90	50.63	31.59	4.58
16	41.62	53.14	80.94	84.07	89.80	74.78	56.24	37.58	6.10
17	30.20	37.63	57.01	60.13	62.28	69.51	55.51	23.46	3.26
18	23.57	25.59	41.75	46.15	45.86	41.45	61.97	33.37	2.33
19	8.86	8.68	12.45	13.13	14.33	12.27	13.69	44.11	1.23
prop	67.18	78.03	87.70	90.15	92.39	86.83	83.95	71.60	12.50

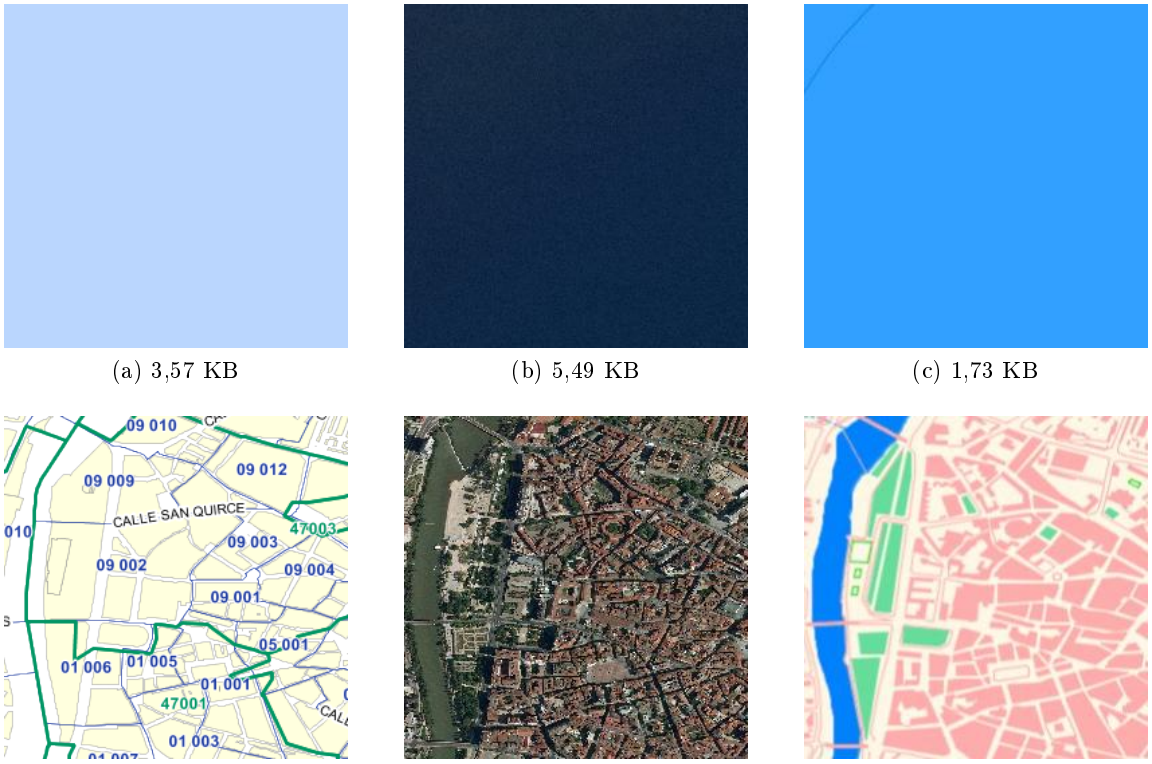
partir de las estadísticas recogidas en aquel identificado por la fila, de acuerdo con el modelo simplificado. La última columna muestra el consumo de recursos, medido como porcentaje de teselas cacheadas por el modelo. Así, la celda sombreada en la Tabla 7.1 indica que, utilizando como fuente de predicción las estadísticas recogidas para el nivel de resolución 13, se consigue un porcentaje de aciertos de caché en el nivel de resolución 18 de 91.95 %, para lo que se requiere cachear el 11.97 % de los objetos.

Sin embargo, el beneficio del uso de una caché parcial no reside en el ahorro en el número de teselas cacheadas, sino en el espacio de almacenamiento y el tiempo de generación ahorrados. Como se expone en (Quinn y Gahegan, 2010) y se ha verificado en el presente

Tabla 7.3: Porcentaje (%) de aciertos de caché mediante el modelo simplificado obtenido con los registros de PNOA, con μ =media.

	12	13	14	15	16	17	18	19	recursos
12	21.10	25.17	27.47	18.88	20.30	32.07	11.36	12.74	0.64
13	11.20	22.82	21.83	17.17	30.66	29.32	11.58	13.18	0.25
14	6.60	15.92	21.45	17.88	28.14	23.29	10.59	12.22	0.17
15	3.45	9.93	15.67	17.85	29.45	18.81	8.43	9.70	0.11
16	2.40	6.53	11.17	12.45	28.00	13.46	6.16	7.10	0.06
17	0.95	4.57	7.52	9.63	12.64	5.94	4.58	5.20	0.03
18	45.25	64.98	73.61	75.58	86.04	89.34	86.49	88.61	9.06
19	39.40	61.72	69.89	67.10	72.86	69.04	82.54	85.22	7.71
prop	40.05	55.97	64.05	61.22	72.28	64.53	76.77	80.26	8.93

estudio, el ahorro en el número de teselas cacheadas supera al conseguido en términos de espacio de almacenamiento. Estos resultados revelan que las teselas de mayor "interés" (las más solicitadas) incurren en un mayor coste de almacenamiento que aquellas omitidas.



(a) 3,57 KB (b) 5,49 KB (c) 1,73 KB

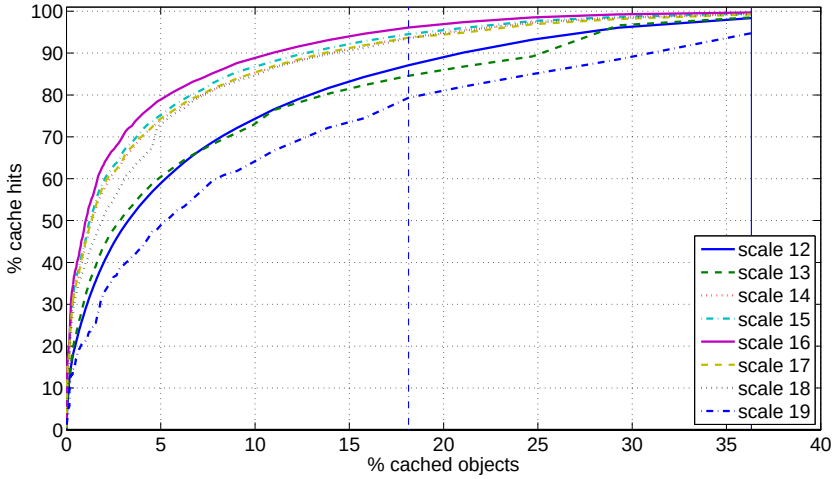
(d) 101,08 KB (e) 29,95 KB (f) 12,56 KB

Figura 7.8: Espacio de almacenamiento de una tesela para Cartociudad (izquierda), PNOA (centro) e IDEE-Base (derecha), cubriendo una zona de mar (arriba) y una urbana (abajo).

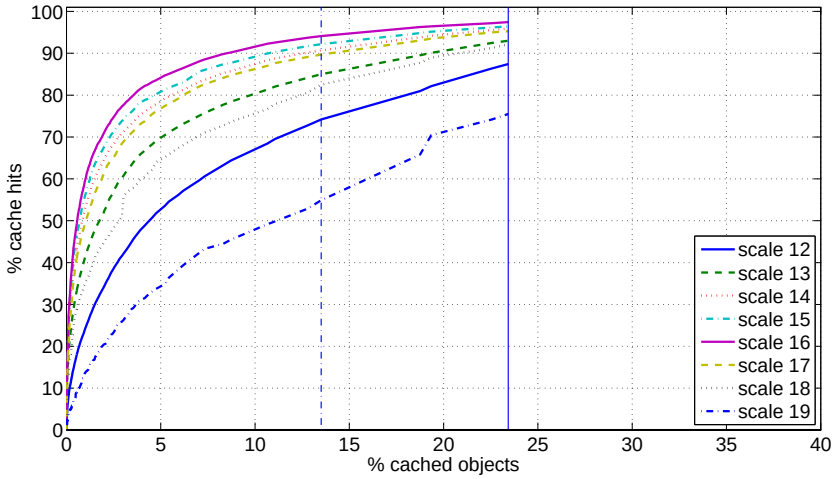
A modo de ejemplo, se muestra en la Figura 7.8 la gran diferencia entre el espacio de almacenamiento en disco ocupado por una tesela que recibe un elevado número de peticiones (una zona urbana) y otra que no se ha solicitado (una zona de mar). Este efecto aplica incluso en mayor proporción en el tiempo de creación de las teselas, tal y como se indica en (Quinn y Gahegan, 2010).

En la Figura 7.7 se representa el porcentaje de aciertos conseguidos por el modelo frente al porcentaje de objetos cacheados para el servicio IDEE-Base. La línea vertical punteada identifica el umbral de probabilidad ($\mu = E\left\{\hat{P}\right\}$). A partir de un determinado porcentaje de objetos cacheados, identificado por una línea vertical continua, el modelo no es capaz de realizar ninguna predicción. Los objetos restantes (fracción de objetos a la derecha de dicha línea) corresponden a teselas que no han sido solicitadas y por tanto no figuran peticiones de las mismas en los *logs*.

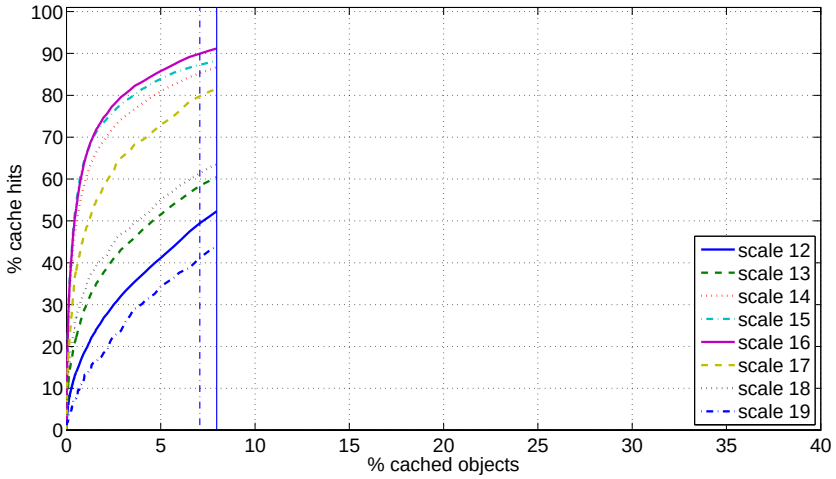
Como se puede observar, el modelo simplificado ofrece buenos resultados cuando, como fuente de la predicción para un determinado nivel de resolución, se utiliza otro nivel de resolución cercano, empeorando cuando éstos son distantes. Al descender en la pirámide de escalas hacia niveles de resolución más fina, el porcentaje de objetos solicitados se reduce, por lo que el rango en el que el modelo es capaz de realizar predicciones también lo hace.



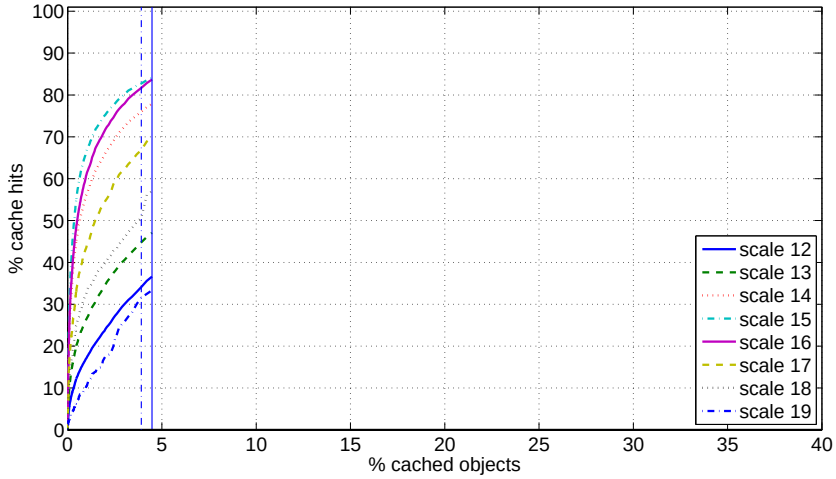
(a) Escala 12



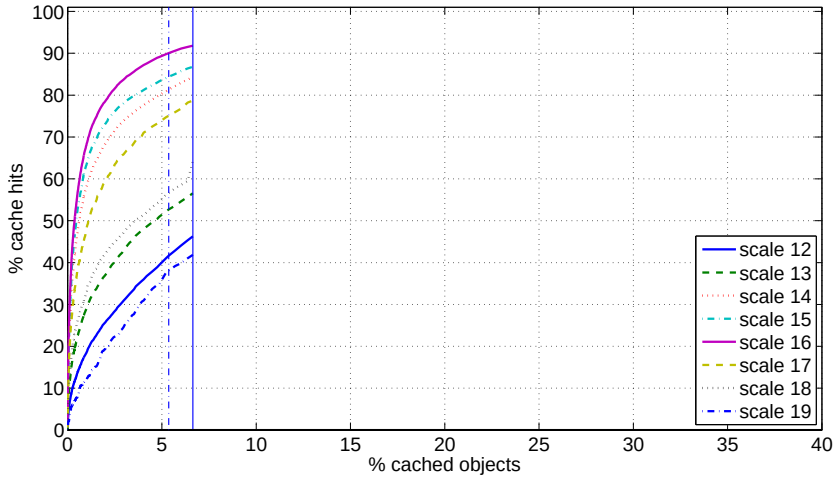
(b) Escala 13



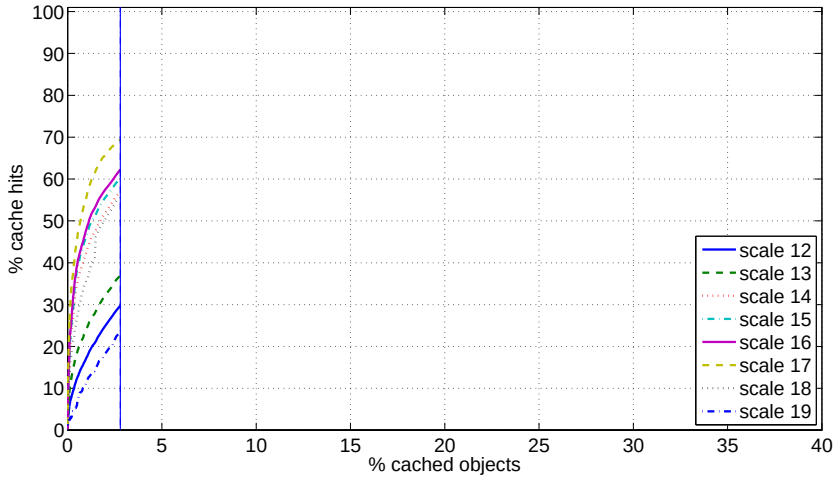
(c) Escala 14



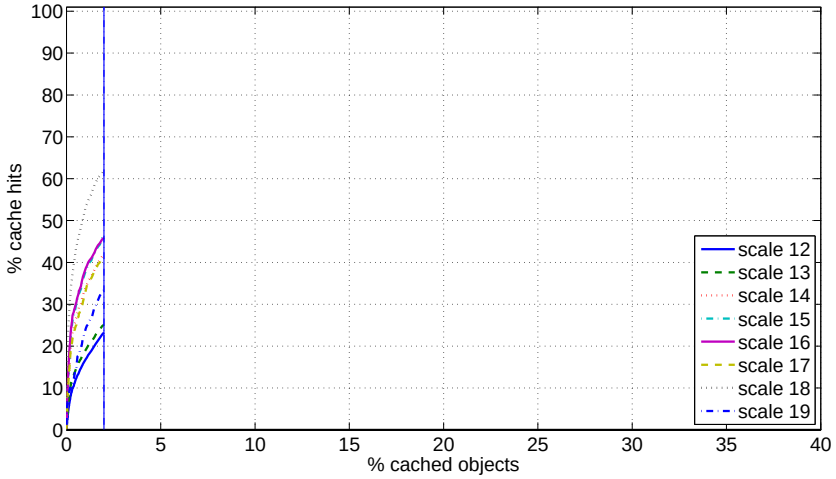
(d) Escala 15



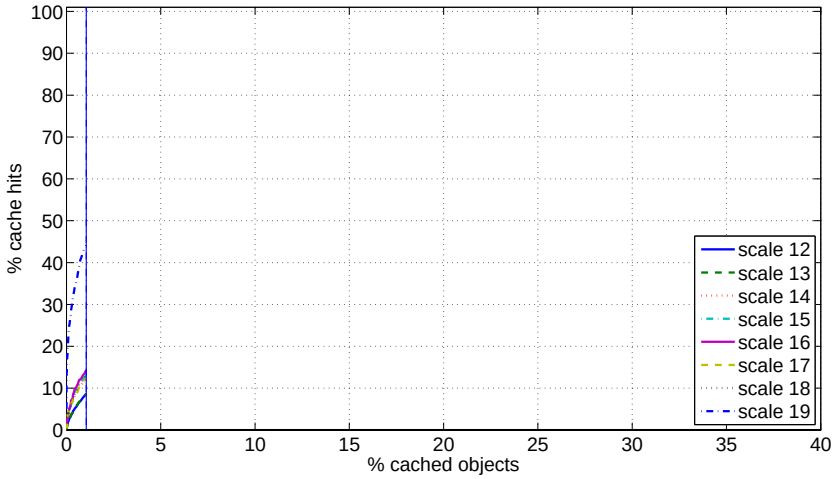
(e) Escala 16



(f) Escala 17



(g) Escala 18



(h) Escala 19

Figura 7.7: % Aciertos de caché Vs % de objetos cacheados en IDEE.

También se ha evaluado un modelo que combina ponderadamente todos los fenómenos (las estadísticas recogidas en otros niveles), para estimar los accesos en un determinado nivel. A partir de los registros normalizados de peticiones, se calculan los pesos de la combinación lineal de cada uno de estos mapas de estimación probabilística de forma que se minimice el error cuadrático medio con el mapa de estimación probabilística objetivo.

En las Tablas 7.4 y 7.5 se muestran las *lambdas* o coeficientes de esta combinación lineal, y los porcentajes de aciertos en caché y de objetos *cacheados*, respectivamente. Se deduce que pueden obtenerse tasas de acierto elevadas, manteniendo un reducido consumo de recursos, sin conocimiento previo de las estadísticas de un determinado nivel, simplemente considerando aquellas recogidas para otras escalas.

Tabla 7.4: Coeficientes de la combinación lineal que minimiza el error cuadrático medio para la predicción de las peticiones de los usuarios en un determinado nivel a partir de las estadísticas recogidas en otros niveles en IDEE-Base.

	$\lambda 12$	$\lambda 13$	$\lambda 14$	$\lambda 15$	$\lambda 16$	$\lambda 17$	$\lambda 18$	$\lambda 19$
12	0	0.0202	0.1101	-0.0321	0.0870	0.0462	0.0264	0.0176
13	0.6533	0	0.1264	0.0062	0.0836	0.0313	-0.0085	-0.0056
14	0.2858	0.0104	0	0.4103	0.2503	0.0272	-0.0021	-0.0054
15	-0.0154	-0.0001	0.6950	0	0.2107	-0.0076	0.0244	0.0023
16	0.3627	0.0090	0.4870	0.1788	0	0.1056	0.0066	-0.0084
17	0.6670	0.0115	0.0890	-0.0608	0.3904	0	0.0560	-0.0075
18	0.8980	-0.0058	-0.1127	0.1522	0.0340	0.1026	0	0.0497
19	0.8072	-0.0071	-0.0901	0.0157	0.0046	-0.0504	0.0794	0

Tabla 7.5: Tasa de acierto de caché y consumo de recursos utilizando una combinación lineal que minimiza el error cuadrático medio (LSM) para la predicción de las peticiones de los usuarios para cada nivel a partir de las estadísticas recogidas en los otros niveles en IDEE-Base.

	12	13	14	15	16	17	18	19
% aciertos	58.0545	83.4969	89.0939	87.7049	93.8743	92.8810	91.8991	80.0117
% recursos	8.3050	16.7408	9.7492	7.1892	11.0367	14.7983	16.3608	19.9800

7.3. Modelo Predictivo

Los contenidos del presente apartado han sido parcialmente publicados en (García et al., 2013) y (García et al., 2012a).

A diferencia de los modelos descriptivos, que indican «dónde han estado mirando los usuarios», los modelos predictivos buscan anticipar «dónde van a mirar» en un futuro. El uso de estos últimos frente a los primeros presenta varias ventajas. Una ventaja importante es que los modelos predictivos pueden tener en cuenta tanto la presencia de regiones de mapa que son populares de forma permanente, como las que lo son tan sólo de forma temporal.

Por ejemplo, entre los lugares de popularidad fija más comúnmente solicitados por los usuarios se encuentran los centros urbanos, autopistas y zonas costeras (Fisher, 2007a). Puede anticiparse con frecuencia que estas zonas recibirán un mayor número de peticiones que las zonas adyacentes. Estos lugares de popularidad permanente se predicen con relativa facilidad y figuran con claridad en los modelos descriptivos.

Sin embargo, existen zonas que no presentan un patrón estacionario de peticiones, sino que reciben un elevado número de ellas tan sólo en ciertos periodos de tiempo, como zonas en las que se han producido desastres naturales, zonas que experimentan tensiones políticas, etc. Un modelo descriptivo podría reflejar estos lugares de popularidad temporal únicamente si los registros recogidos se limitan a un corto intervalo de tiempo que incluya los períodos de alta actividad. Sin embargo, estas zonas desaparecerían al considerar las estadísticas de uso a lo largo de un largo período de tiempo.

Por el contrario, los modelos predictivos se basan en la identificación de fenómenos geográficos "potencialmente interesantes" para los usuarios. Estos fenómenos pueden extraerse

a partir de las observaciones obtenidas de un modelo descriptivo, pero también pueden predecirse de forma anticipada sin la necesidad de observaciones previas. Por ejemplo, ante la presencia de un evento de masas programado, como podría ser la celebración de unos juegos olímpicos, puede anticiparse que ciertos fenómenos, como por ejemplo los pabellones deportivos olímpicos y sus zonas adyacentes en este caso, recabarán un mayor interés por parte de los usuarios.

Sin embargo, puede haber casos en los que la identificación de los elementos de interés no resulte una tarea sencilla. En estos casos pueden utilizarse los índices de Moran y Geary para cuantificar la dependencia espacial entre las peticiones de los usuarios y de éstas respecto a determinados fenómenos geográficos o *features*.

7.3.1. Metodología

En la arquitectura propuesta (ver Figura 7.1), el *ModelExtractor* necesita un conocimiento previo de los accesos pasados para poder "aprender" qué colecciones de fenómenos pueden ser buenos indicadores del comportamiento de los usuarios. Esto implica que, durante la puesta en marcha del servicio, existe un período de tiempo inicial que debe esperarse para que se recojan estadísticas suficientes antes de que se pueda ejecutar este mecanismo.

El método propuesto se basa en las siguientes hipótesis:

H1 Existe una correlación espacial entre las peticiones de los usuarios y el contenido semántico de los elementos de la caché. Por tanto, se pueden modelar estas relaciones mediante ciertos fenómenos geográficos, dando lugar a una base vectorial adecuada en la que proyectar los patrones de peticiones observados.

H2 Puede conseguirse una ganancia de rendimiento significativa con un consumo modesto de recursos mediante una combinación de los atributos de ciertos fenómenos geográficos.

H3 Los patrones de acceso del servicio son estacionarios en el marco temporal de aplicación del modelo. Por tanto, un modelo calculado en el presente puede caracterizar fielmente las peticiones en el futuro.

Siempre que se cumplan estas hipótesis, se puede definir una tarea de precarga de caché tras registrar la actividad del sistema durante un determinado período de tiempo. Este modelo predictivo puede usarse para anticipar qué teselas serán pedidas por los usuarios. De esta forma, pueden precargarse las regiones de mapa de mayor popularidad y así mejorar la calidad del servicio y la experiencia del usuario. Del mismo modo, una política de reemplazo de caché podría usar este mismo modelo para eliminar las teselas menos "populares" y así dejar espacio para almacenar nuevos objetos.

El estimador resultante obtiene, para cada tesela, una medida de la probabilidad de acceso futuro. Las teselas se ordenan por orden descendente de probabilidad y aquellas con mayor probabilidad de ser pedidas se introducen en la caché hasta que se alcance el umbral de consumo de recursos deseado.

7.3.1.1. Fuentes de datos

Las simulaciones se han llevado a cabo utilizando los registros de peticiones de los servicios de mapas IDEE-Base y PNOA. Los registros seleccionados suponen un total de

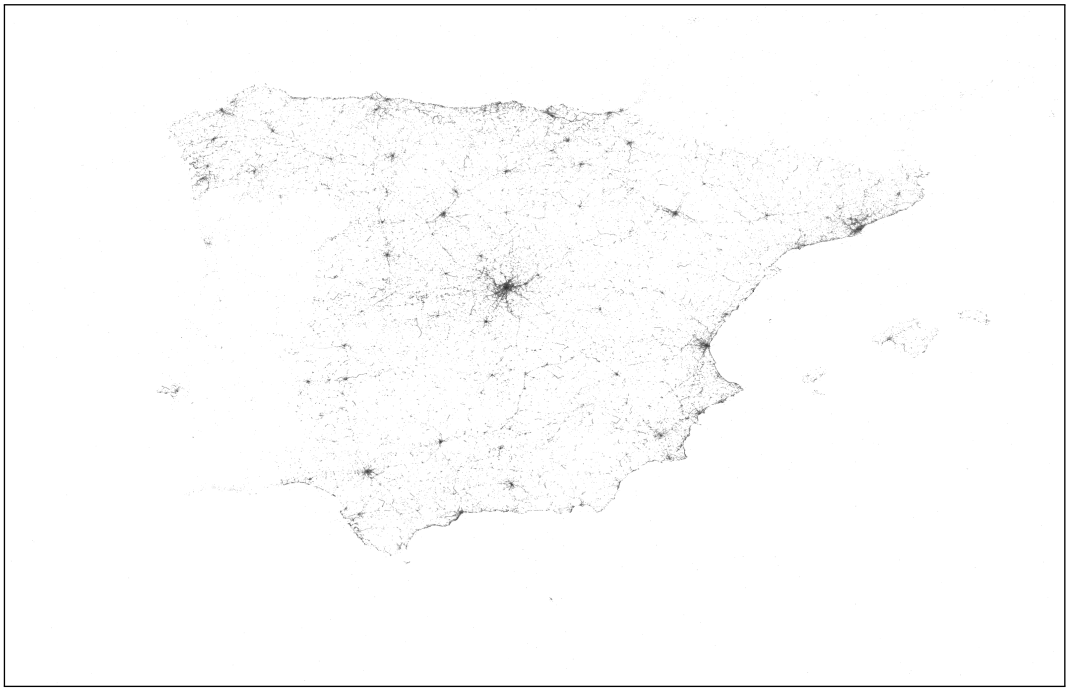


Figura 7.8: Mapa de calor de las peticiones recibidas en el área de estudio por el servicio PNOA en el nivel de resolución 18, entre el 19 de Marzo de 2010 y el 17 de Junio del mismo año. Lás zonas más oscuras corresponden a las teselas que han recibido un mayor número de solicitudes por parte de los usuarios. Para mayor claridad, la representación se realiza en escala logarítmica.

6.5 millones de peticiones recopilados durante 3 meses, entre el 19 de Marzo de 2010 y el 17 de Junio del mismo año.

En la Figura 7.8 se muestra el *heatmap* de peticiones realizadas al servicio de mapas del PNOA al nivel de resolución 18. En esta figura se advierte la presencia de algunas zonas populares asociadas a fenómenos geográficos.

Se ha limitado el área de estudio a la región correspondiente a la Península Ibérica y zonas adyacentes, concretamente al encuadre geográfico definido por las coordenadas geográficas [-11.9971 °W, 5.5804 °E, 33.8379 °S, 45.0872 °N].

Este área de estudio abarca un total de 3200×2048 teselas en el nivel de resolución 15 con los esquemas de teselado usados en IDEE-Base y PNOA, y el número de teselas crece en potencias de cuatro al hacerlo el nivel de resolución.

7.3.2. Modelo predictivo basado en regresiones lineales

A continuación se describe una particularización de la arquitectura presentada en la Sección 7.1. En esta arquitectura concreta se aplica un método regresivo para ajustar las estadísticas recogidas en accesos previos mediante los múltiples fenómenos geográficos del catálogo, usados como variables independientes o predictoras. Como resultados, se obtiene una colección de coeficientes de la regresión lineal que indican el peso de la contribución de cada fenómeno al resultado final del estimador. En el Anexo A.3 se describe una implementación de esta estrategia desarrollada sobre la popular caché de teselas GeoWebCache.

7.3.2.1. Análisis regresivo

El término *análisis regresivo* se utiliza frecuentemente para describir una familia de técnicas que persiguen modelar las relaciones entre una variable dependiente (también llamada variable de respuesta), y un determinado número de variables independientes o predictoras.

Dada una variable $y = \{y_i\}$ con un conjunto de n observaciones o medidas de las variable de respuesta, y sus correspondientes valores del conjunto de m variables independientes $x = \{x_{ij}\}$, se puede construir una serie de ecuaciones lineales como se muestra en (7.8):

$$y_i = \sum_{j=1}^{m} \beta_j x_{ij} + \epsilon_i \tag{7.8}$$

que puede re-escribirse en notación matricial como:

$$Y = BX + E \tag{7.9}$$

Los coeficientes $\beta = \{\beta_j\}$ de la regresión lineal indican el peso de cada variable en la combinación lineal. Dado que el número de observaciones n es generalmente mayor que el de coeficientes m (se dice que el sistema es "sobredeterminado"), se requiere añadir una componente de error adicional ϵ para permitir un grado de ajuste del modelo con los valores observados.

Una aproximación a este problema consiste en buscar una mejor solución para el vector β que minimice la diferencia entre el modelo ajustado y los valores observados. En la práctica, el procedimiento más ampliamente utilizado, denominado *Ordinary Least Squares* (OLS), busca minimizar la suma del error cuadrático medio. Mediante esta aproximación, la solución para los coeficientes se obtiene según la siguiente expresión matricial:

$$\hat{B} = (X^T X)^{-1} X^T Y \tag{7.10}$$

donde el símbolo ($\hat{}$) indica que se trata de una estimación.

7.3.2.2. Definición del modelo

En este trabajo, cada observación y_i de la variable dependiente corresponde al número normalizado de peticiones de cada tesela de mapa. Las variables independientes X corresponden a los fenómenos geográficos, cuyos valores se extraen a partir de atributos contenidos o asociados a un fenómeno.

En este trabajo, las variables se han definido en el dominio del espacio de teselado como una función de la distancia de cada tesela al fenómeno geográfico más cercano. Esta aproximación difiere de la adoptada en (Quinn y Gahegan, 2010), donde se aplica un *buffer* en torno a las geometrías de los fenómenos de interés. Este *buffer* pretende reflejar la correlación espacial entre las peticiones de las teselas englobadas dentro del encuadre geográfico de la pantalla, mientras los usuarios se desplazan a lo largo del mapa. De esta forma, no sólo se cachean las teselas que intersectan con los fenómenos de interés, sino también aquellas otras teselas que se encuentran dentro de este *buffer*. Este procedimiento da lugar a resultados *booleanos*. Por el contrario, aquí se aplica una función de decaimiento gausiana $f(d)$ (ver expresión (7.11)) en torno a las geometrías de los fenómenos geográficos. Con ello se pretende modelar las siguientes suposiciones:

- Las teselas que recaban un mayor interés por parte de los usuarios son aquellas que contienen o son atravesadas por uno de estos fenómenos geográficos.

- Las teselas vecinas que caen dentro de la vista del mapa también son de interés para los usuarios.

- La correlación semántica cae abruptamente con la distancia d, medida en número de teselas.

$$f(d) = e^{-\frac{1}{2}(\frac{d}{bw})^2} \tag{7.11}$$

El parámetro bw controla la anchura de la gausiana. El valor de este parámetro está relacionado con el número de teselas que caen generalmente dentro del área visible de los clientes de mapa. Durante las simulaciones se ha escogido un valor de 10 teselas para este parámetro ($bw = 10$).

De este modo, las variables quedan definidas en el rango [0,1] y no es necesario aplicar ninguna normalización adicional.

7.3.2.3. Evaluación del modelo

Para validar las hipótesis establecidas en la Sección 7.3.1 frente al modelo propuesto, se han utilizado los registros de peticiones o *logs* procedentes de los servicios de mapas WMS-C del PNOA e IDEE-Base. Al tratarse de trazas registradas en un sistema real, estos *logs* representan un patrón del comportamiento de los usuarios más realista que si se tratase de un patrón sintético.

Los registros de peticiones se han dividido en dos intervalos temporales. El primero de ellos se utiliza como fuente para "entrenar" el modelo del predictor, mientras que el segundo se usa para validar las predicciones realizadas con el modelo creado previamente.

El primer conjunto de datos emula las peticiones recibidas tras el arranque del servicio. Esto se corresponde con la etapa de entrenamiento. Durante esta etapa se recogen y actualizan los contadores de peticiones de las teselas. Al finalizar esta etapa de entrenamiento, se aplica un análisis regresivo de tipo OLS para obtener el mejor ajuste de las variables extraídas del catálogo de fenómenos con las observaciones de este primer conjunto de datos. Los coeficientes de la combinación lineal resultantes de la regresión OLS se utilizan en el predictor para estimar la "popularidad" futura de las teselas. El segundo conjunto de datos simula las peticiones futuras y se utiliza para validar el modelo predictivo.

7.3.2.4. Simplificación del modelo

Como se ha comentado al describir la arquitectura genérica propuesta, se ha recurrido a una simplificación del modelo para reducir la sobrecarga derivada del mismo. Concretamente, los experimentos se han llevado a cabo simplificando el modelo de acuerdo a la rejilla definida por el nivel de resolución 15. El modelo se utiliza para realizar predicciones sobre los niveles de resolución 16 y 18 respectivamente, para IDEE-Base y PNOA, utilizando como entrenamiento las peticiones recibidas por estos servicios en dichas escalas.

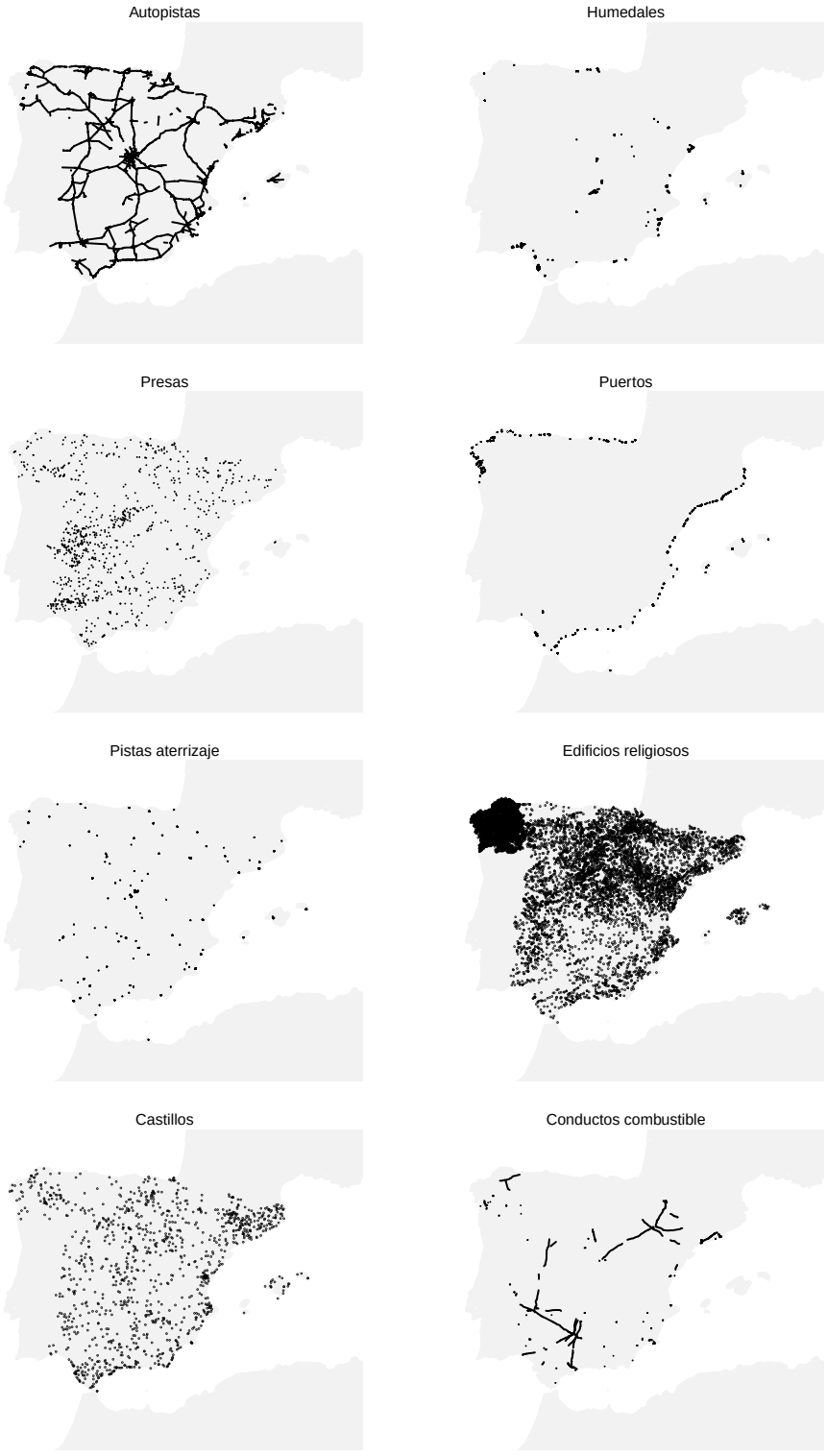


Figura 7.9: Catálogo de fenómenos geográficos utilizado para simulación del modelo regresivo.

Tabla 7.6: Fenómenos geográficos del catálogo, con sus respectivos tipos de geometría y el porcentaje de teselas cubiertos por cada fenómeno al nivel de resolución 15.

Tipo de fenómeno	*Geometría*	*Cobertura (%)*
Autopistas	Multilinestring	0.4832
Castillos	Point	0.0169
Edificios religiosos	Point	0.1612
Pistas aterrizaje	Multipolygon	0.0149
Puertos	Multipolygon	0.0221
Humedales	Multipolygon	0.0399
Presas	Multilinestring	0.0217
Conductos combustible	Multilinestring	0.0831

7.3.2.5. Catálogo de fenómenos geográficos

Las variables explicativas o regresoras se calculan para cada tesela de mapa utilizando diversas fuentes de datos geográficos extraídos de la Base Cartográfica Numérica (BCN200) del IGN, un sistema de información geográfica multipropósito que alberga datos topográficos y temáticos. En la Tabla 7.6 se recoge una breve descripción de cada uno de los tipos de fenómenos utilizados, con su tipo de geometría, así como el porcentaje de teselas cubiertas en la zona de estudio por cada fenómeno. Estos fenómenos se ilustran en la Figura 7.9 sobre un mapa para su interpretación visual. En este estudio, en lugar de utilizar un selector de fenómenos para la extracción automática de los fenómenos más adecuados, éstos han sido seleccionados manualmente. Se han considerado ambos tipos de fenómenos, tanto aquellos que intuitivamente se esperaba fuesen mas relevantes al modelo, como aquellos con previsiones de resultar irrelevantes.

7.3.2.6. Resultados y discusión

Se aplica el método regresivo OLS para ajustar las variables independientes (atributos de los fenómenos geográficos) a la variable dependiente (conjunto de datos de entrenamiento con las peticiones de los usuarios).

Los conjuntos de datos de entrenamiento incluyen las peticiones recibidas en cada servicio desde el 19 de Marzo de 2010 y para distintos intervalos de tiempo: 1 hora, 5 horas, 1 día, 7 días y 30 días. Esto permite analizar la estacionariedad de los resultados en función del tiempo de entrenamiento. El conjunto de datos usado para la validación de las predicciones incluye las peticiones extraídas de los *logs* desde el 19 de Abril al 17 de Junio de 2010 ($\sim$ 2 meses). En la Tabla 7.7 se resume toda la información anterior.

Las Figuras 7.10 y 7.11 muestran la evolución de los coeficientes de la regresión lineal a lo largo del tiempo para los servicios PNOA e IDEE-Base, respectivamente. Los resultados muestran que la influencia de cada fenómeno en el modelo es distinta para cada servicio. Por ejemplo, mientras que en PNOA los fenómenos de tipo *Puerto* tienen un mayor peso que las de tipo *Pista de aterrizaje*, en el caso de IDEE-Base sucede lo contrario. Los pesos negativos identifican variables que podríamos interpretar como "repulsoras", de forma que compensen el efecto de otras con influencia positiva en regiones no deseadas.

Tabla 7.7: Número de peticiones registradas en los conjuntos de datos usados para simulación en PNOA e IDEE en los niveles de resolución 18 y 16, respectivamente.

	Entrenamiento					*Validación*
	1 hora	*5 horas*	*1 día*	*7 días*	*30 días*	*62 días*
$PNOA_{18}$	1029	1965	50478	323482	847058	4861289
$IDEE_{16}$	382	728	10073	78316	379778	696743

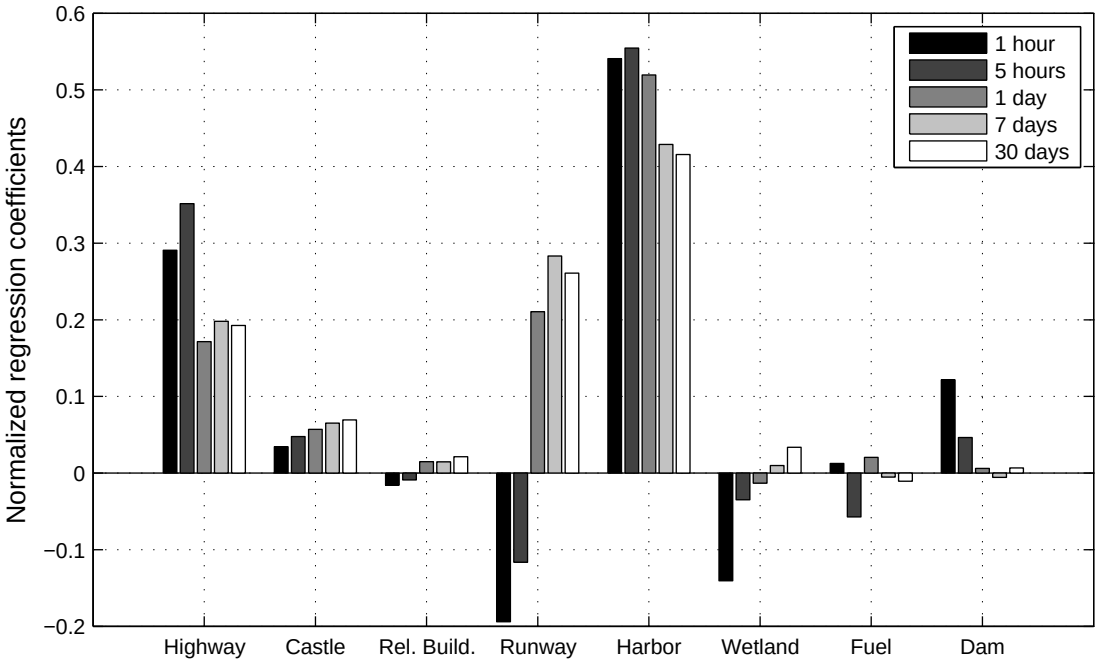


Figura 7.10: Coeficientes de la regresión OLS en PNOA, para diferentes períodos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. Los coeficientes están normalizados de forma que la suma de los coeficientes positivos es igual a 1.

El modelo ha sido validado experimentalmente mediante simulaciones siguiendo el algoritmo descrito a continuación:

1. Aquellas teselas que son pedidas durante la etapa de entrenamiento son insertadas en la caché.

2. El modelo predictivo se define como la combinación lineal de las entradas con los pesos calculados durante la etapa de entrenamiento. Este modelo realiza una estimación de la popularidad de las teselas, que se usa entonces para la definición de prioridades para la política de precarga de teselas.

3. La tarea de precarga se simula mediante un procesamiento exhaustivo de los datos para asignar una estimación de probabilidad a cada objeto.

4. Las teselas se ordenan por orden descendente de probabilidad emulando la tarea de

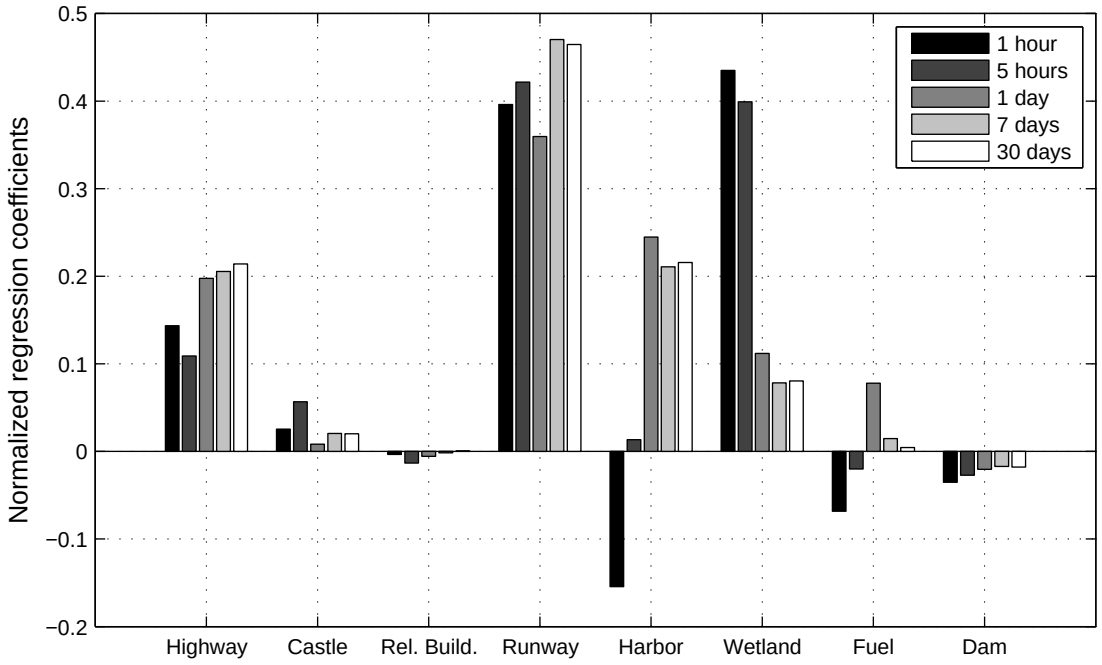


Figura 7.11: Coeficientes de la regresión OLS en IDEE-Base, para diferentes períodos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días. Los coeficientes están normalizados de forma que la suma de los coeficientes positivos es igual a 1.

precarga.

5. El contenido de la caché se modela como un subconjunto de este grupo ordenado cuyo tamaño depende del porcentaje de caché deseado. Con el objetivo de evitar la aparición de sesgos debidos a la disposición espacial de las teselas, aquellas con el mismo nivel de prioridad se escogen de forma aleatoria.

En las Figuras 7.12 y 7.13 se muestra el rendimiento conseguido respectivamente en PNOA e IDEE-Base, medido como la tasa de acierto de caché (CHR - *Cache-Hit Ratio*), obtenido precargando primero las teselas que se estima serán pedidas con mayor frecuencia, y precargando progresivamente el resto de las teselas por orden descendente de prioridad hasta que se alcanza el límite de consumo de recursos establecido.

Con el objetivo de evaluar el rendimiento del modelo propuesto (etiquetado como *ols* en las figuras), se muestra una comparación con:

- Un algoritmo óptimo (*opt*), no implementable en la práctica, ya que usa el conocimiento pleno de las peticiones futuras, pero que resulta útil para realizar un análisis comparativo y conocer el máximo rendimiento que se puede obtener.
- Una estrategia que mantiene en caché las teselas pedidas durante la etapa de entrenamiento y que selecciona de forma aleatoria el resto de los objetos para introducirlos en la caché (*hist*).

Ambos métodos, *ols* y *hist*, coinciden durante una pequeña fracción inicial de las teselas ($\sim 1\%$). Estos aciertos de caché corresponden a los conseguidos por las teselas pedidas y cacheadas durante la etapa de entrenamiento, antes de la aplicación de heurísticas. En

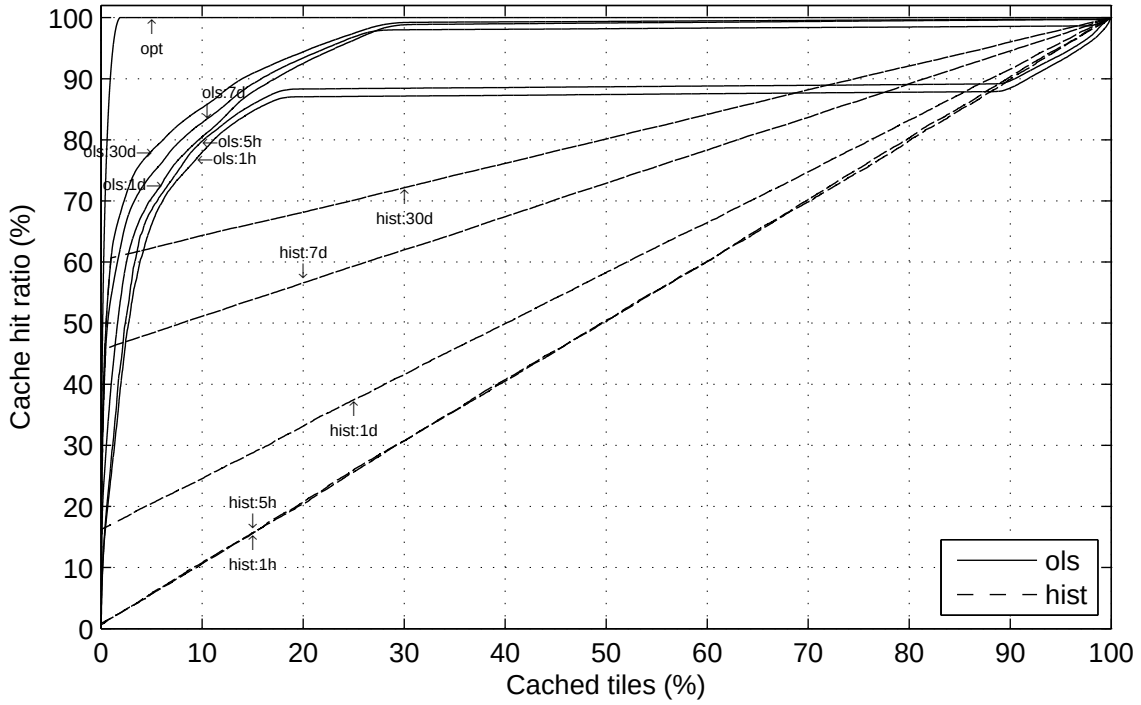


Figura 7.12: Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Cache-Hit Ratio (CHR), obtenidas en PNOA para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días.

ambos servicios, PNOA e IDEE-Base, los aciertos de caché conseguidos por estos objetos aumenta al hacerlo el tiempo de entrenamiento. Esto se debe a la estacionariedad de las peticiones; durante un tiempo más amplio se recibe un mayor número de peticiones, y éstas (y sus vecinas) se piden de nuevo en el futuro con bastante probabilidad.

A partir de esta fracción inicial de los objetos, donde ambos métodos coinciden, el método *ols* rápidamente supera el rendimiento del método *hist* a medida que aumenta la proporción de teselas cacheadas. Como el método *hist* basa su funcionamiento exclusivamente en el conocimiento de los accesos pasados, éste no es capaz de definir prioridades más allá de los objetos cacheados, por lo que cachea el resto de las teselas de forma aleatoria. Por el contrario, el método *ols* es capaz de generalizar y realizar predicciones sobre el resto de los objetos; Para ello hace uso de las relaciones geográficas implícitas entre las variables para determinar qué fenómenos geográficos recaban el mayor interés por parte de los usuarios.

Por otra parte, el método *ols* puede alcanzar tasas de acierto elevadas en menor tiempo que el método *hist*, puesto que este último requiere un tiempo muy elevado para recibir un número suficiente de peticiones que sea capaz de reflejar los patrones a más largo plazo.

Tal y como se muestra en la Figura 7.12, en PNOA se alcanza el estado estacionario muy pronto, no obteniéndose mejoras significativas al incrementar el tiempo de entrenamiento por encima de un día. Sin embargo, en IDEE-Base (ver Figura 7.13) se consigue un refinamiento del modelo aumentando el tiempo de entrenamiento, dando lugar a mejoras progresivas en el rendimiento conseguido. En cualquier caso, el modelo consigue tasas de acierto superiores al 90 % con periodos de entrenamiento breves de unas pocas horas y con

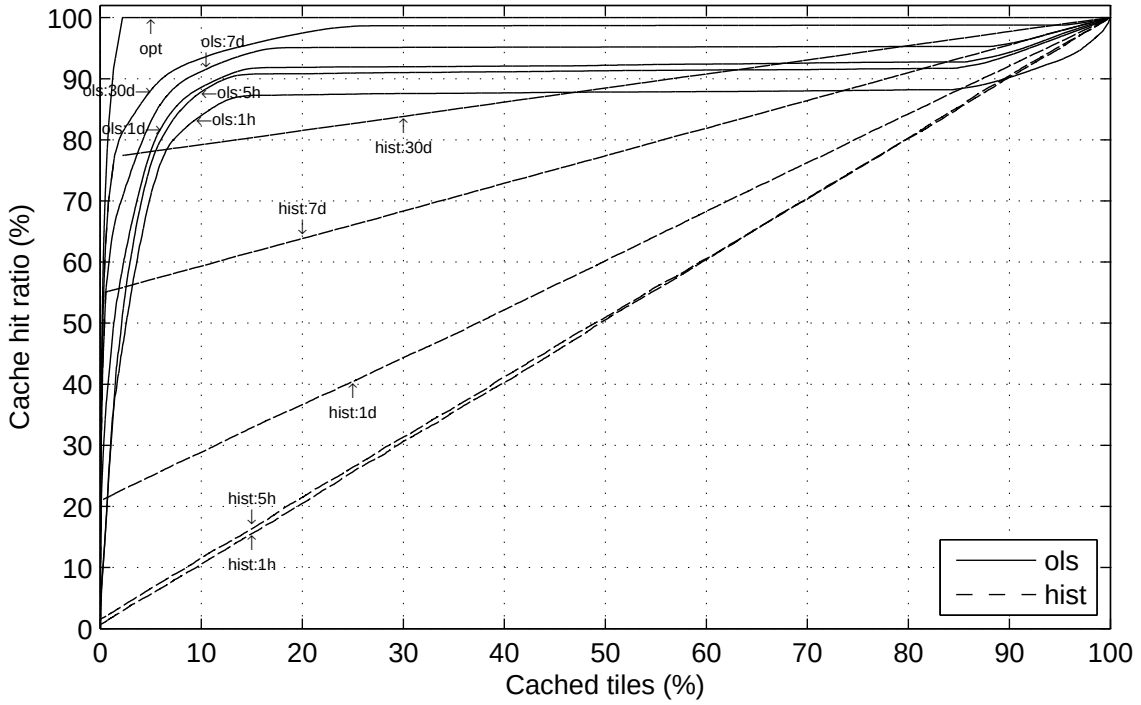


Figura 7.13: Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Cache-Hit Ratio (CHR), obtenidas en IDEE-Base para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días.

una reducida fracción de objetos cacheados inferior al 20 %.

De esta forma, en PNOA, para tiempos de entrenamiento de un día, y para una fracción de objetos cacheados del 20 %, el método *ols* obtiene una tasa de acierto de caché (CHR - *Cache-Hit Ratio*) del 92.49 %, mientras que el método *hist* obtiene una tasa de acierto de tan sólo el 32.84 %. En cuanto a IDEE, manteniendo el mismo uso de recursos y tiempo de entrenamiento, el método *ols* consigue una tasa de acierto del 91.86 %, frente al 36.89 % obtenido por el método *hist*.

En las Figuras 7.14 y 7.15 se muestran los resultados de rendimiento obtenidos, medidos como el *Byte-Hit Ratio* (BHR), para PNOA e IDEE-Base, respectivamente. Como se puede observar, las tasas de acierto BHR conseguidas tanto por el método *ols* como por la estrategia *hist*, superan en todo momento a los obtenidos en términos de CHR. Tal y como se argumenta en (Quinn y Gahegan, 2010), las zonas populares tienen una mayor variación de patrones y colores, lo que hace que las teselas, almacenadas como ficheros de imágen comprimidos, requieran una mayor cantidad de espacio en disco que aquellas teselas con menos cantidad de elementos gráficos. Dado que las teselas más populares se cachean en primer lugar, y generalmente son de mayor tamaño, se obtienen mejores resultados con esta métrica que con el CHR (*Cache-Hit Ratio*). En términos cuantitativos, en PNOA, para un tiempo de entrenamiento de un día, y para una fracción de objetos cacheados del 20 %, el método *ols* obtiene una tasa de BHR del 97.56 %, mientras que el método *hist* obtiene una tasa de tan sólo el 42.91 %. En el caso de IDEE-Base, manteniendo el mismo uso de recursos y tiempo de entrenamiento, el método *ols* consigue un BHR de 97.12 % frente al 46.09 % obtenido por el método *hist*.

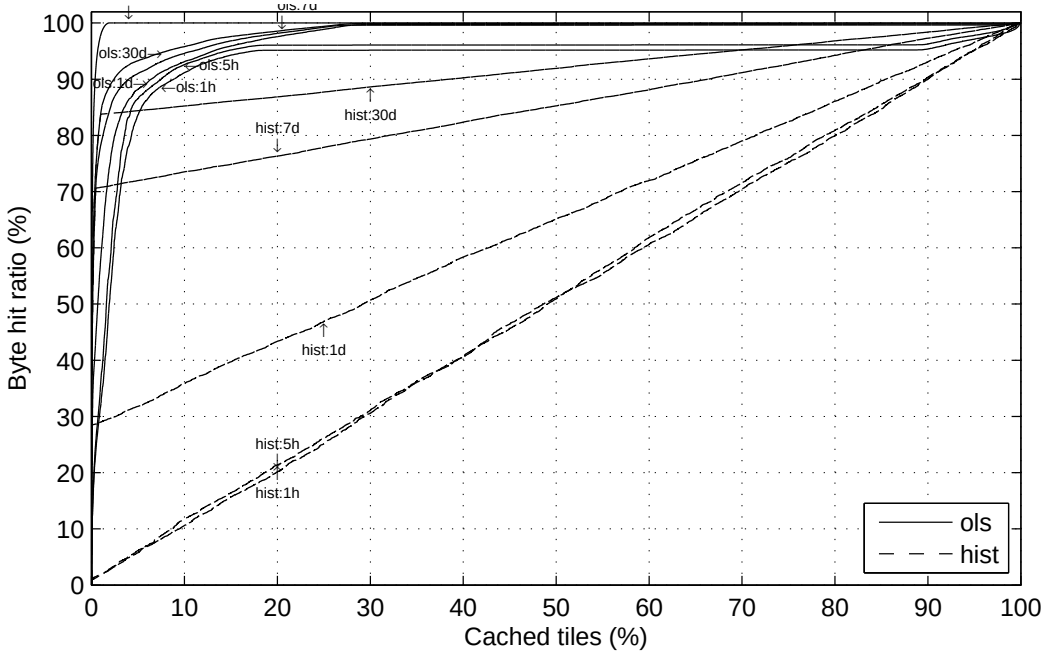


Figura 7.14: Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Byte-Hit Ratio (BHR), obtenidas en PNOA para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días.

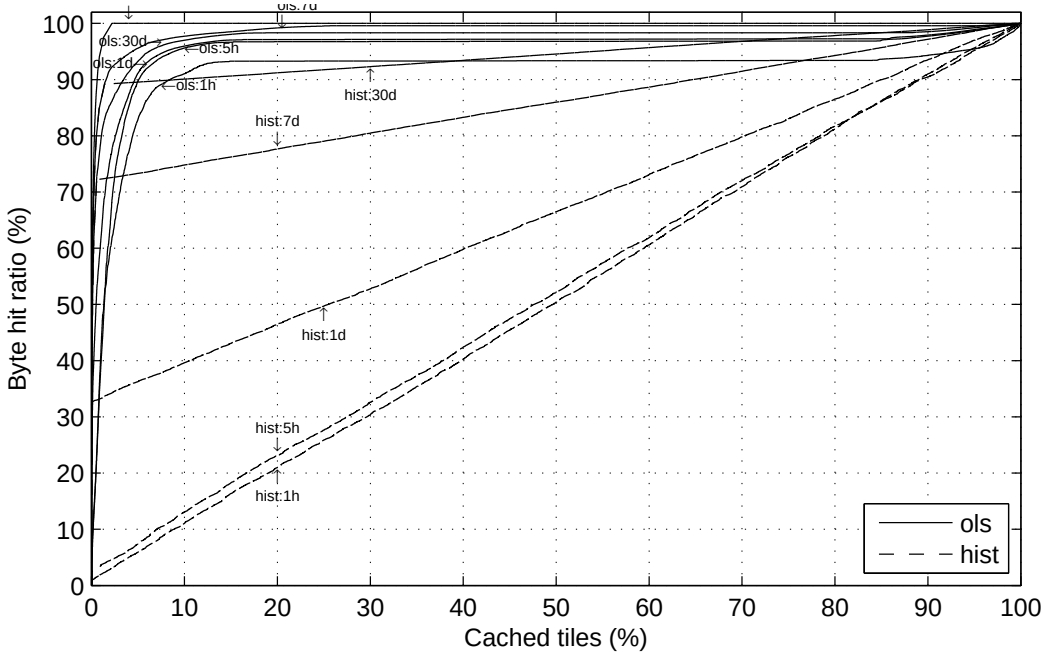


Figura 7.15: Medidas de rendimiento comparativas para los modelos *ols* y *hist*, en términos de Byte-Hit Ratio (BHR), obtenidas en IDEE-Base para distintos tiempos de entrenamiento: 1 hora, 5 horas, 1 día, 7 días y 30 días.

7.3.3. Modelo predictivo basado en redes neuronales

Además del modelo regresivo comentado anteriormente, se ha realizado una implementación de la arquitectura presentada en la Sección 7.1 (ilustrada en la Figura 7.1) mediante un sistema inteligente basado en redes neuronales. A continuación se detallan los bloques específicos para esta implementación concreta.

7.3.3.1. Extractor del modelo

En este caso, el extractor del modelo (*ModelExtractor*) utiliza un mecanismo de Aprendizaje Automático (*Machine Learning*) para predecir la popularidad futura de las teselas a partir de los fenómenos geográficos seleccionados del catálogo. Para formalizar el concepto de "aprendizaje" empleamos la convención establecida por Mitchell (1997):

> *"Se dice que un programa aprende de la experiencia E con respecto a cierta clase de tareas T y una medida del rendimiento P si la medida del rendimiento de tareas en T medidas por P aumenta con la experiencia E."*

En este contexto:

- Tarea T: predecir la popularidad de las teselas.
- Medida del rendimiento P: error medio entre la popularidad estimada y la popularidad real extraída de las observaciones de la historia del servicio.
- Experiencia de entrenamiento E: una secuencia de teselas con sus respectivos atributos de las *features* y el número de peticiones recibido por cada tesela normalizado respecto al máximo número de peticiones.

Concretamente, en este estudio se han empleado redes neuronales artificiales (ANN) como técnica de aprendizaje automático. Las redes ANN ya fueron introducidas en el Capítulo 6.1.

Entrenar una neurona artificial consiste en elegir los valores de los pesos de forma que se obtenga la salida deseada para las entradas introducidas. Esta fase de entrenamiento o de aprendizaje puede segregarse como "aprendizaje supervisado" (requiere un conjunto de datos de entrada cuya respuesta objetivo se conoce de antemano) o "aprendizaje no supervisado" (no se indica la solución al problema). En el sistema propuesto, los pesos de la red neuronal se ajustan mediante aprendizaje supervisado utilizando tuplas de la forma $\langle \overline{x}, t \rangle$.

Los ejemplos de entrenamiento $\overline{x}$ consisten en información asociada a cada tesela o grupo de teselas. La salida supervisada t para cada registro de entrenamiento corresponde al número normalizado de peticiones recibido por dicha tesela o grupo de ellas de acuerdo a los *logs* registrados por el servidor.

Así, un registro de entrenamiento de la red neuronal para una determinada tesela podría ser de la forma siguiente:

$\langle \langle x_{autopista} = 1, x_{puerto} = 0{,}4, \ldots, x_{humedal} = 0{,}6 \rangle, t = 0{,}9 \rangle$

En este trabajo se ha utilizado una clase especial de red neuronal conocida como *Multi-Layer Perceptron* (MLP), en la que todas las neuronas de cada capa se conectan con con todas las neuronas de la capa anterior.

A pesar de que el clásico algoritmo de retropropagación ó *backpropagation* ha sido la técnica de aprendizaje de redes neuronales más comúnmente utilizada, aquí hemos utilizado el algoritmo Levenberg-Marquardt (Marquardt, 1963) con el objetivo de acelerar el aprendizaje. Este algoritmo es uno de los más rápidos encontrado en la literatura, pudiendo resultar entre 10-100 veces más rápido que el sencillo algoritmo de gradiente descendente (Hagan y Menhaj, 1994).

7.3.3.2. Planificador de la tarea de precarga

Los parámetros de salida del *Model Extractor* son los valores de los pesos $w_{i,j}$ de la red neuronal. El planificador de la tarea de precarga (*seeding planner*) utiliza estos pesos para evaluar la red neuronal para cada tesela, dando lugar a un *raster* cuyos valores asignan diferentes probabilidades a las teselas. Este planificador de precarga alimenta la caché con las teselas precargadas de acuerdo a las prioridades asignadas, y por orden descendente de probabilidad.

7.3.3.3. Catálogo de fenómenos geográficos

El catálogo de fenómenos se ha poblado utilizando 27 tipos de fenómenos, extraídos de la Base Cartográfica Numérica (BCN200) del IGN. Sin embargo, el selector de fenómenos (*Feature Selector*) selecciona aquellos fenómenos que no intersectan con el área de estudio, y descarta los más inter-correlados (aquellos con valores de correlación cruzada por encima del 90 %). Esto reduce el catálogo a tan sólo 19 fenómenos, con una mínima pérdida de información.

7.3.3.4. Fuentes de datos

Las simulaciones se han llevado a cabo utilizando los registros de acceso del PNOA. Estos registros suman aproximadamente un total de 5.5 millones de peticiones recibidas a lo largo de tres meses, entre el 19 de Marzo de 2010 y el 17 de Junio del mismo año.

A partir de estos datos, los registros de entrenamiento incluyen un total de 48,671 peticiones recibidas durante el 19 Marzo de 2010. Por otra parte, el registro de datos de validación utilizado para evaluar las predicciones realizadas engloba un total de 5,385,074 peticiones, registradas en los *logs* entre el 20 de Marzo y el 17 de Junio de 2010 ($\cong$ 3 meses).

7.3.3.5. Simulación

Se han generado y probado múltiples configuraciones de red neuronal, variando la arquitectura de la misma. En el modelo propuesto se han testeado tanto redes neuronales de una única capa oculta como de dos, variando el número de neuronas de cada capa oculta entre 3 y 10. Finalmente, se ha seleccionado la configuración que ha ofrecido los mejores resultados de entre todas las configuraciones evaluadas.

La arquitectura final seleccionada tiene una capa de entrada con 19 entradas (una para cada tipo de *feature*), una única capa oculta compuesta de 6 neuronas y una sola salida (ver Figura 7.16). De acuerdo a la convención habitual, esta configuración puede etiquetarse como una red de tipo 19/6/1. La función de transferencia utilizada en la capa oculta es de tipo *tangente-sigmoidea*, la cual realiza un escalado no-lineal pasando de un rango infinito

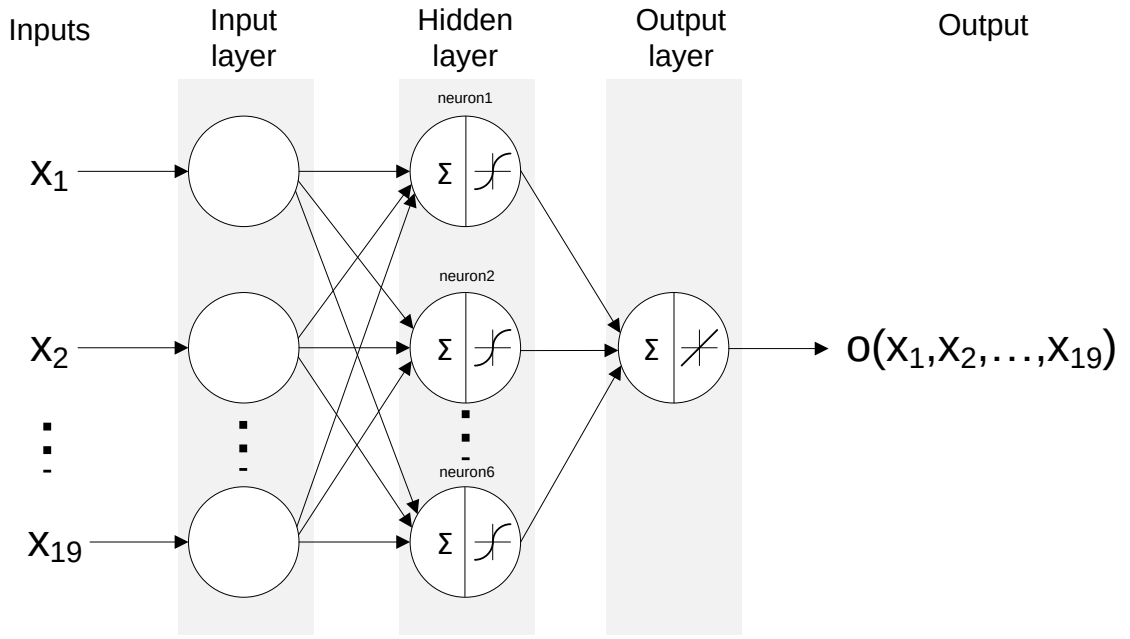


Figura 7.16: Arquitectura de ANN de tipo 19/6/1 utilizada en este estudio. La función de activación de las neuronas de la capa oculta es de tipo tangente-sigmoidea, mientras que en la capa de salida se utiliza una función lineal.

de valores al intervalo $(-1, 1)$. Por su parte, en la capa de salida se utiliza una función de transferencia lineal.

Al igual que en el modelo regresivo, las variables de entrada asociadas a cada tesela han sido definidas en función de la distancia de ésta al fenómeno geográfico más cercano, aplicando la función de suavizado gausiana (7.11), con $bw = 10$.

Tal y como se hizo en el Capítulo 6.1, se han normalizado las entradas de forma que todos los valores entren en el intervalo $[-1, 1]$, utilizando el escalado (6.1).

La red neuronal se entrena mediante aprendizaje supervisado utilizando los registros de peticiones del servicio PNOA, normalizando la salida objetivo al intervalo [0,1]. Inicialmente, los registros de datos se dividieron en tres conjuntos: entrenamiento, validación y test. Estos conjuntos de datos contienen las peticiones recibidas durante el primer día, segundo día, y peticiones registradas a partir del segundo día, respectivamente. El primero de ellos se utiliza para entrenar la red neuronal. El segundo se utiliza para validar que la red neuronal mantiene la habilidad de generalizar los resultados de forma adecuada y para identificar la aparición del *overfitting*. El último conjunto de datos se usa como un test completamente independiente para comprobar la generalización de la red neuronal.

Un inconveniente propio de las redes neuronales es que éstas son susceptibles de sufrir lo que se conoce como *overfitting* o sobre-entrenamiento. Si la red neuronal se entrena durante demasiadas iteraciones, puede aprender detalles irrelevantes presentes en los datos de entrenamiento, lo cuál conduce a una mala generalización. Por lo tanto, un aspecto crítico a tener en cuenta a la hora de entrenar la red neuronal reside en la elección de las condiciones de parada del algoritmo (Hopfield y Tank, 1985).

En esta línea, se ha utilizado un conjunto de datos de validación para detectar la apa-

Tabla 7.8: Parámetros de la red neuronal propuesta.

Parámetro	Valor
Arquitectura	Feed-forward Multilayer Perceptron
Entradas	19
Capas ocultas	1
Neuronas en cada capa oculta	6
Salida	1
Funciones de activación	Tangente-sigmoidea en la capa oculta, lineal en la capa de salida
Función de error	Error cuadrático medio (mse)
Algoritmo de entrenamiento	Levenberg-Marquardt
Método de aprendizaje	Aprendizaje supervisado
Modo de actualización de los pesos	Batch mode
Condición de parada	Alcance de un gradiente de error mínimo

rición de *overfitting*. Sin embargo, durante los experimentos se ha comprobado que el error de generalización, medido para los ejemplos de validación, decrece monótonamente con el entrenamiento, al igual que ocurre con el error de los datos de entrenamiento, por lo que esta condición de parada no se alcanza nunca. Por esta razón, no se ha hecho uso del registro de datos de validación, reduciendo el tiempo de puesta en marcha necesario para obtener estos datos adicionales. Los resultados ponen de manifiesto que la curva de error de los datos de entrenamiento se aplana rápidamente, por lo que el rendimiento apenas mejora con el entrenamiento adicional. Por ello, el entrenamiento se detiene cuando se alcanza un gradiente mínimo del error de 10^{-5}. Los detalles de la red neuronal utilizada en este experimento se recogen en la Tabla 7.8.

Los experimentos han sido llevados a cabo simplificando el modelo de acuerdo a la rejilla definida por el nivel de resolución 14. El modelo se utiliza para realizar predicciones sobre el nivel de resolución 18

7.3.3.6. Resultados

La red neuronal descrita previamente ha sido entrenada para ajustar las variables explicativas o predictoras (atributos de los fenómenos geográficos) a la variable dependiente (registro de datos de entrenamiento con las peticiones de los usuarios). Durante las simulaciones, la red neuronal ha experimentado una convergencia muy rápida, produciéndose la condición de parada al alcanzar el gradiente de error mínimo en tan solo 33 iteraciones.

En la Figura 7.17 se muestran los resultados de rendimiento conseguido, medidas en términos de tasa de acierto de caché (CHR). Estos resultados se han obtenido precargando primero las teselas que se esperaba fuesen pedidas con mayor frecuencia y precargando progresivamente el resto de teselas por orden descendente de prioridad hasta que se alcanza el límite de consumo de recursos establecido.

Con el objetivo de evaluar el rendimiento del método propuesto (etiquetado como `nnet` en la figura), éste ha sido comparado con:

- Un algoritmo óptimo (`best` u `opt`) no implementable en la práctica.

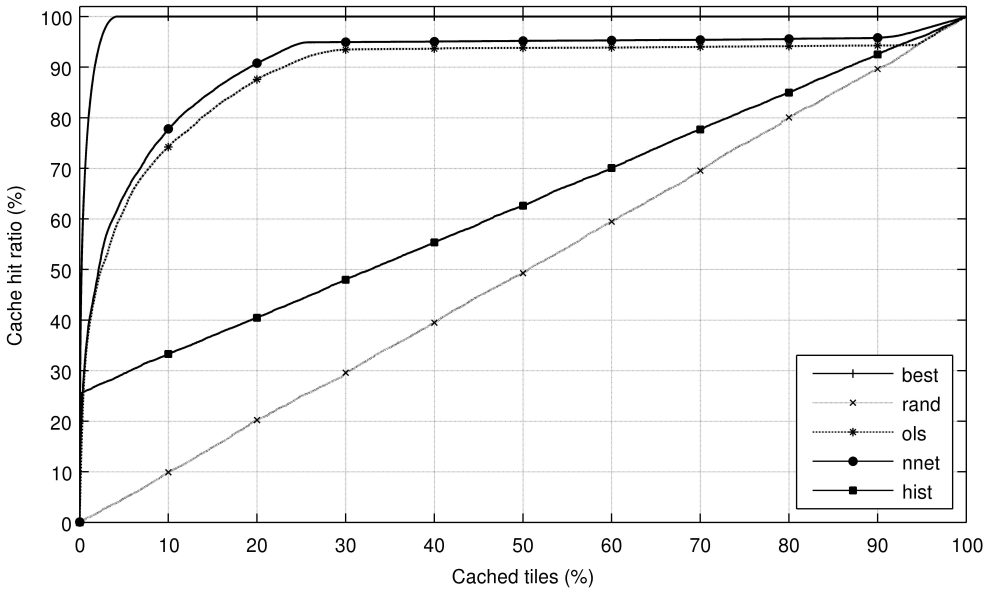


Figura 7.17: Comparativa de rendimiento conseguido en PNOA en términos de *Cache-Hit Ratio* para un tiempo de entrenamiento de 1 día.

- Una estrategia aleatoria (`rand`) que selecciona aleatoriamente objetos para cachear.
- Una estrategia que mantiene el histórico de los pasados accesos y cachea las teselas que han sido pedidas con anterioridad por orden descendente de popularidad (`hist`).
- El modelo predictivo que asigna prioridades a las teselas de forma automática como una combinación lineal de los fenómenos geográficos del catálogo utilizando un estimador regresivo por mínimos cuadrados para ajustar el modelo a los accesos pasados (`ols`), descrito en la Sección 7.3.2.

El algoritmo óptimo muestra que las peticiones de los usuarios se concentran en menos de un 5 % del total de teselas presentes en el espacio de teselado definido por la zona de estudio. Esta curva identifica el máximo rendimiento que se puede obtener.

Comparando el rendimiento de la estrategia `hist` frente a las basadas en *features* (`ols` and `nnet`), se observan resultados interesantes. Inicialmente, para una reducida fracción de teselas cacheadas, el rendimiento de la estrategia `hist` supera al conseguido por estas dos. Sin embargo, ambas `ols` y `nnet` rápidamente superan el rendimiento de la estrategia `hist` a medida que aumenta la proporción de teselas cacheadas. La acusada pendiente inicial de la curva de la estrategia `hist` corresponde al rendimiento conseguido cacheando las teselas que han sido solicitadas previamente durante la fase de entrenamiento, las cuáles reciben un elevado número de peticiones a continuación. Sin embargo, dado que el método `hist` basa su funcionamiento exclusivamente en el conocimiento de accesos pasados, este método no es capaz de definir prioridades a los objetos más allá de esta pequeña fracción de las teselas, por lo que toma decisiones de precarga de forma aleatoria. Por el contrario, utilizando las relaciones geográficas implícitas, las estrategias basadas en fenómenos son capaces de generalizar y realizar predicciones sobre qué combinación de fenómenos geográficos recaba

el mayor interés por parte de los usuarios.

Atendiendo a las estrategias basadas en *features*, se observa que se obtienen predicciones muy acertadas a largo plazo (≈ 3 meses) entrenando estos modelos durante un breve período de tiempo (1 día). Siempre que estos modelos obtengan una tasa de acierto elevada en las predicciones realizadas en base a una combinación de fenómenos geográficos, las hipótesis H1 y H2 (ver apartado 7.3.1) pueden considerarse verdaderas. Por otra parte, la elevada precisión obtenida en las predicciones realizadas a largo plazo apoyan el supuesto inicial de la hipótesis H3.

Hay una parte plana en ambas curvas en la que la tasa de acierto de caché no mejora con la incorporación de nuevos objetos a la caché, ya que estos objetos no se piden a continuación. Por tanto, en esta región, las estrategias basadas en *features* predicen accesos futuros que no se llegan a producir. Por el contrario, la parte final de la curva corresponde a teselas que erróneamente no se espera que fueran pedidas.

Comparando ambas estrategias basadas en el uso de *features*, se ha observado que `nnet` obtiene mejor rendimiento que `ols`. Mientras que el método `ols` únicamente puede producir relaciones lineales de sus *features* de entrada, la propuesta basada en redes neuronales puede aprender funciones no-lineales arbitrarias. Sin embargo, en comparación con *Multiple Linear Regression* (MLR), las redes neuronales tienen el inconveniente de presentar una capacidad limitada de identificar explícitamente relaciones causales. Esto dificulta la interpretación de los pesos de la red neuronal, que actúa como una "caja negra". Debido a esto, no permite determinar de forma sencilla qué variables contribuyen en mayor medida a una salida concreta. Además, el modelado de la red neuronal requiere un mayor consumo de recursos computacionales.

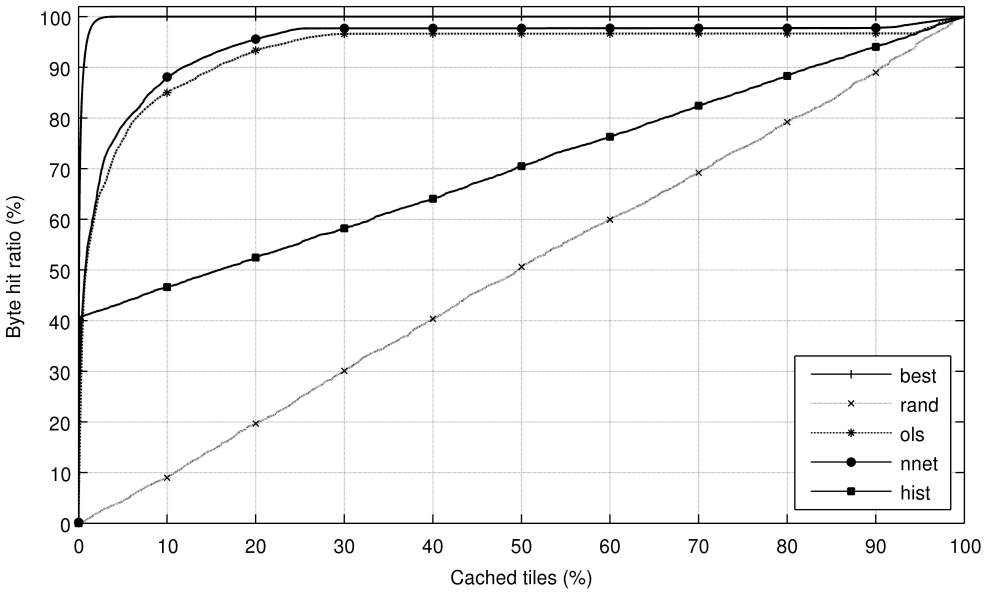


Figura 7.18: Comparativa de rendimiento conseguido en PNOA en términos de *Byte-Hit Ratio* para un tiempo de entrenamiento de 1 día.

En la Figura 7.18 se muestra una comparativa de rendimiento, medido según el BHR. Las medidas de BHR obtenidas con todas las estrategias, exceptuando la estrategia aleatoria

`rand`, superan consistentemente aquellas obtenidas para el CHR. La justificación de este hecho ya fue comentada en el Apartado 7.3.2.6.

7.4. Análisis comparativo frente a los sistemas de gestión de carga dinámica

Con el objetivo de valorar correctamente los resultados obtenidos mediante los mecanismos de precarga propuestos en esta investigación, se presenta a continuación la evolución temporal de la tasa de acierto y consumo de recursos de una caché de teselas que utilice un sistema de carga dinámica en el que la caché se vaya poblando a medida que los usuarios van realizando las peticiones. Este es el comportamiento habitual de las cachés implementadas en la práctica.

Para simular el procedimiento de carga dinámica, se ha hecho uso de los registros del servicio PNOA, en los que cada petición de mapa lleva asociado un *timestamp* con el tiempo en el que se realizó cada petición.

En la Figura 7.19 se muestra la evolución temporal de la tasa de acierto de caché (*Cache Hit Ratio*) y del consumo de recursos (porcentaje de objetos cacheados) conseguidas en PNOA mediante el procedimiento de carga dinámica.

Se ha simulado un tiempo de funcionamiento de carga dinámica en la caché de un mes, con las peticiones recibidas entre el 19 de Marzo al 19 de Abril de 2010. Las tasas de acierto de caché han sido evaluadas frente a los registros comprendidos entre el 19 de Abril de 2010 al 17 de Junio del mismo año.

Estos dos intervalos coinciden con los utilizados para los registros de entrenamiento y validación de los modelos predictivos presentados en la Sección 7.3.

Atendiendo a los resultados presentados en la Figura 7.19, obtenidos con el procedimiento de carga dinámica, se observa que no se obtiene una tasa de acierto de caché del 60 % hasta transcurridos 17 días de funcionamiento, redundando en un 3 % de objetos cacheados.

Esta misma tasa de acierto se consigue con el modelo regresivo presentado en la Sección 7.3.2 (ver Figura 7.12) con un tiempo de entrenamiento de tan solo una hora, y con un consumo de recursos del 6 %. Ese 6 % de objetos se pueden generar mediante el modelo de precarga, utilizando para ello toda la capacidad de cómputo del generador de teselas, en un tiempo significativamente inferior a los 17 días requeridos por el sistema tradicional. Adicionalmente, el predictor permite cubrir otras zonas con alta probabilidad de recibir peticiones futuras y además completar la caché para conseguir tasas de acierto superiores respecto a las peticiones recibidas a más largo plazo.

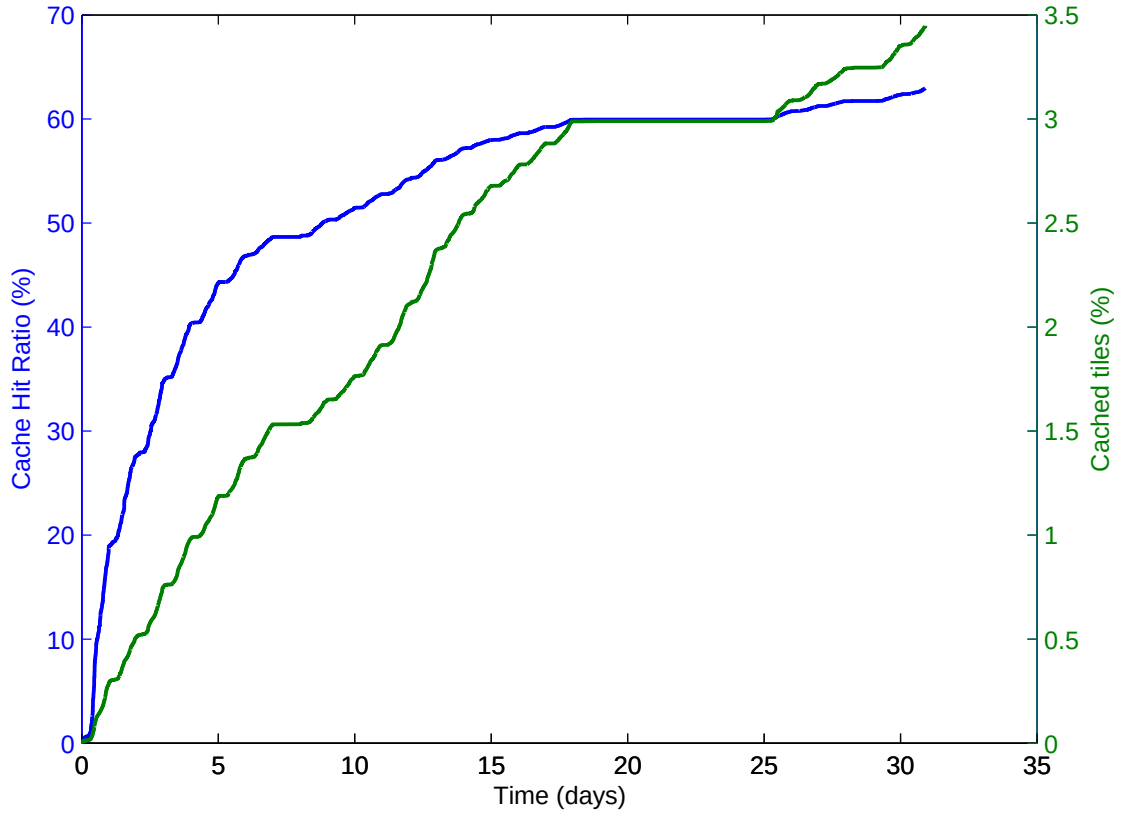


Figura 7.19: Evolución temporal de la tasa de acierto y del consumo de recursos del servicio PNOA mediante el procedimiento de carga dinámica con las peticiones de los usuarios.

Capítulo 8

Políticas de carga dinámica

Resumen: Un importante mecanismo que puede ayudar a mejorar el rendimiento de los servicios WMS-C es la carga dinámica mediante heurísticas. Ante una petición de un usuario, el sistema puede predecir cuál puede ser la siguiente petición o grupo de peticiones. Generando el resultado de estas predicciones dentro del intervalo de tiempo entre dos peticiones sucesivas se puede conseguir una mejora sustancial en la experiencia de usuario, a condición de que las predicciones sean acertadas.

Los contenidos del presente capítulo han sido parcialmente publicados en (García et al., 2011g).

Una línea de investigación insinuada por las expresiones (3.5) y (3.7) es la mejora, mediante estrategias de racionalización de la carga ejecutadas en el *proxy* (módulo *TileGenerator* de la arquitectura presentada en el Capítulo 7.1), del parámetro τ_m. En el proceso de generación de teselas hay diversos cuellos de botella que se pueden mejorar. Una de las partes del proceso de dibujado es el acceso a los almacenes externos de información geográfica. Ya que las consultas suelen estar optimizadas mediante índices espaciales, resulta más eficiente realizar una petición de una tesela de tamaño $n \times n$ (ver Figura 8.1) que n^2 peticiones por separado.

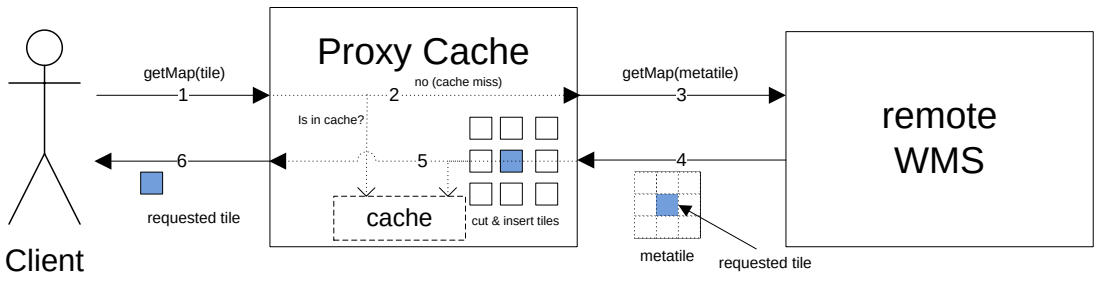
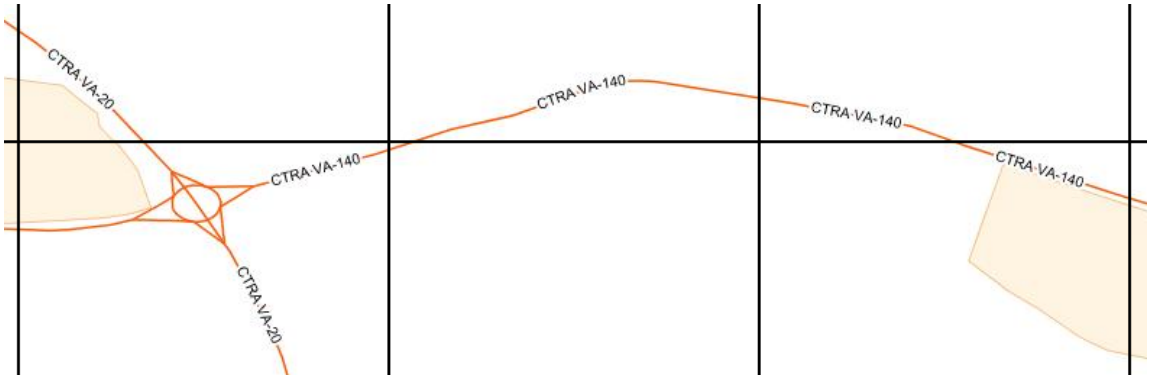
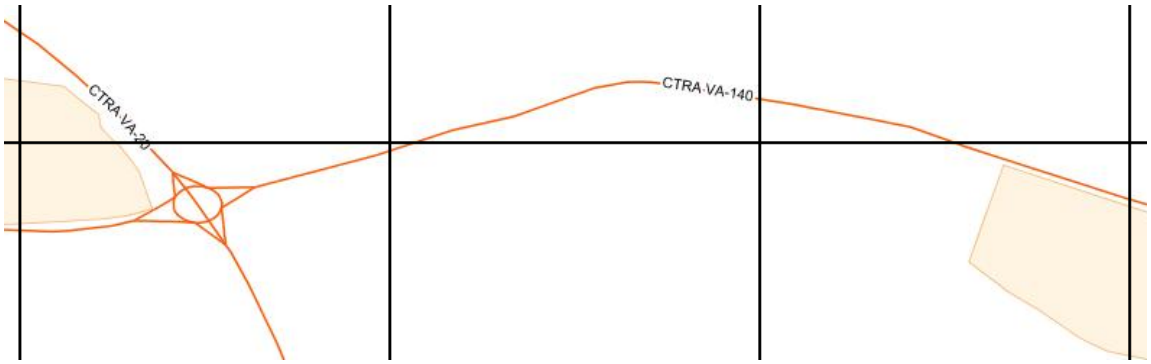


Figura 8.1: Flujo de una petición de mapa con metatiling.



(a) Buffer=0



(b) Buffer=2

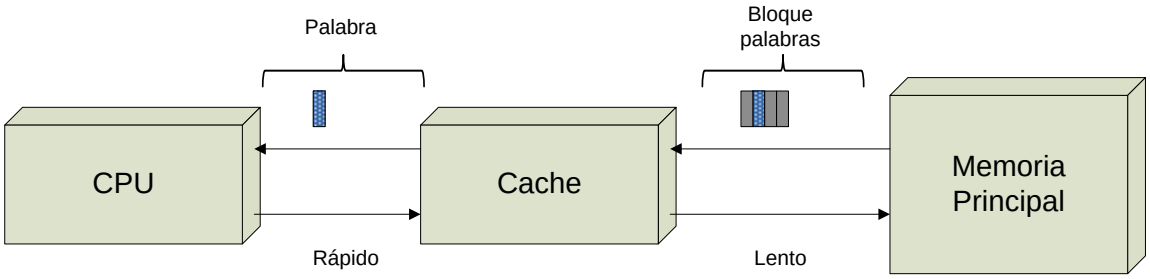
Figura 8.2: Problema del etiquetado redundante en un servicio de mapas. (a) Al pedir teselas individuales aparecen etiquetas duplicadas entre teselas adyacentes. (b) Con metatiling no se producen duplicados de etiquetas.

Otro beneficio obtenido indirectamente es la reducción del problema del etiquetado fraccionado o redundante provocado por el teselado de las peticiones. Se produce cuando un determinado fenómeno, como por ejemplo un río, un lago o una carretera, se extiende a lo largo de múltiples teselas, de forma que el servidor de mapas añade una etiqueta para el mismo fenómeno que se repite en cada una de ellas, tal y como se aprecia en la Figura 8.2.

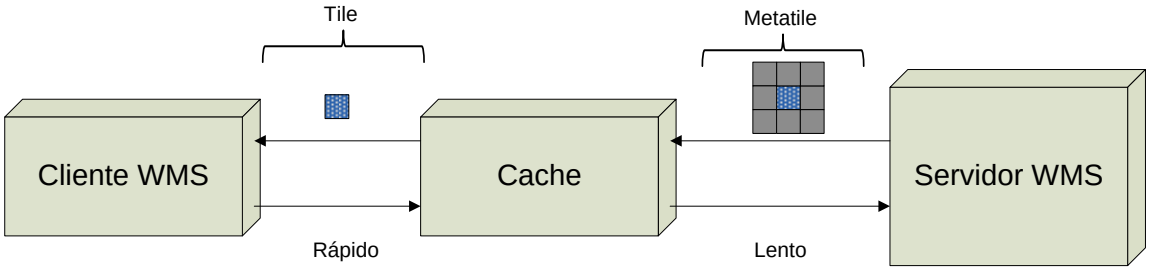
Es por tanto una práctica habitual el generar peticiones de mayor tamaño que la tesela a cachear (esta super-tesela se denomina *metatile*) y posteriormente postprocesarlas para aprovechar la información disponible y generar nuevas teselas. Este procedimiento, conocido como *metatiling*, funciona de la forma siguiente: cuando el *proxy* recibe una petición de una tesela de mapa y se produce un fallo de caché, éste solicita una imagen de mayor tamaño o *metatile* al *backend* remoto. Este *metatile* incluye la tesela solicitada por el cliente, pero también las teselas adyacentes. Entonces, el *proxy* recorta el *metatile* en teselas individuales, devuelve la tesela solicitada al cliente, y almacena todos los fragmentos del *metatile* en la caché. Este procedimiento se ilustra en la Figura 8.1.

Es inmediato establecer una analogía entre el mecanismo de *metatiling* en los servicios de mapas con la transmisión de palabras por bloques en las jerarquías de memoria de los ordenadores (Stallings, 2006). Esta analogía se ilustra en la Figura 8.3.

Las implementaciones investigadas permiten especificar el número adicional de teselas



(a) Transferencia de un bloque de palabras en una jerarquía de memoria (Stallings, 2006).



(b) Transferencia de un *metatile* en un servicio de mapas.

Figura 8.3: Analogía de la transferencia de información por bloques entre una jerarquía memoria de un ordenador y un servicio de mapas.

alrededor de la realmente solicitada (*buffer de B teselas*) que se van a pedir en una sola petición al servidor WMS. De esta manera se le pide al servidor de mapas una *metatile* de tamaño $(2B+1)^2$ teselas centrada en el elemento realmente solicitado.

En un escenario de caché no completa (pero no vacía) esta elección del área a generar no resulta muy eficiente, puesto que es muy probable que algunas de las teselas próximas a la solicitada ya estén disponibles en la caché.

Partiendo de la suposición de que la zona que engloba la tesela solicitada no está homogéneamente cargada, se ha desarrollado un algoritmo para la elección óptima de las *metatiles* a generar. El algoritmo propuesto, ilustrado en la Figura 8.4 busca obtener, en función del estado de la caché, la *metatile* que, conteniendo la tesela solicitada (no necesariamente centrado en la misma), minimice la correlación espacial:

$$R_n(\Delta i, \Delta j) = \sum_{l=i-N}^{j+N} \sum_{m=j-N}^{j+N} h[l+\Delta i, m+\Delta j, n] \tag{8.1}$$

donde la función *booleana* definida en el dominio discreto $h[i,j,n]$ se define según (8.2):

$$h[i,j,n] = \begin{cases} 1 & \textit{si la tesela T(i,j,n) está en la caché} \\ 0 & \textit{en caso contrario} \end{cases} \tag{8.2}$$

Interpretando la correlación en (8.1) como una medida de la semejanza de la información contenida en ambos objetos (*metatile* y caché), parece evidente que la *metatile* que tenga una menor correlación espacial con el estado de la caché es aquella que proporciona la mayor información al sistema, puesto que la información que contiene es complementaria en mayor

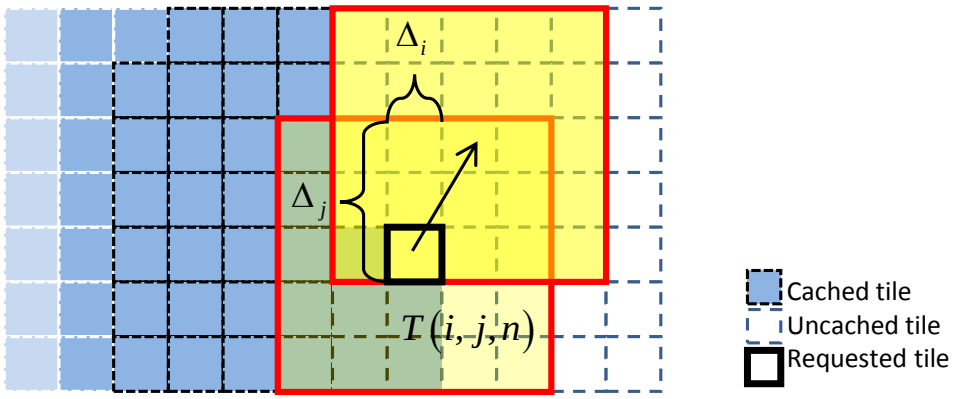


Figura 8.4: Estrategia de mínima correlación con la caché para la generación de *metatiles*.

grado a la ya disponible. En la Figura 8.4 se ilustra la configuración con la que se consigue un mínimo en la redundancia o en la información mutua.

La implementación realizada de esta técnica incluye además un procedimiento en segundo plano para realizar el postproceso de las teselas adicionales. De esta forma se busca minimizar también el tiempo de servicio del *proxy* (τ_h).

Para validar la hipótesis de que puede obtenerse una mejora mediante el uso del *metatiling*, se ha realizado un experimento utilizando el prototipo *WMSCWrapper* (ver Apéndice A.2) como caché de teselas y el servicio de Ocupación del Suelo (CORINE) del IGN como servidor de mapas remoto. Se han realizado 2000 peticiones de mapas distintas al *proxy*, partiendo de una caché inicialmente vacía, para distintos tamaños de *metatile*, y se han analizado las latencias experimentadas en la devolución de las teselas.

Tabla 8.1: Latencias medias para la obtención de un objeto a partir del servicio WMS original (CORINE) en función del tamaño del *metatile*.

buffer (B)	$\tau_{m,metatile}$	$\tau_{m,metatile\_n}$	$G_{metatiling}$
0 (sin metatiling)	1454,10 ms	1454,10 ms	1
1 (metatile 3x3)	2933,94 ms	325,99 ms	4,46
2 (metatile 5x5)	5660,63 ms	226,42 ms	6,42
3 (metatile 7x7)	9561,54 ms	195,13 ms	7,45

Los resultados del experimento se recogen en la Tabla 8.1. La columna $\tau_{m,metatile}$ corresponde a la latencia media de un fallo de caché para los distintos tamaños de *metatile*, que incluye los retardos de transmisión y propagación en la red, el tiempo de generación de la imagen en el servidor de mapas remoto y los tiempos de procesamiento en el *proxy cache* de teselas. Los valores $\tau_{m,metatile\_n}$ se obtienen normalizando aquellos de la columna anterior por el número de teselas que componen el *metatile* ($[2B+1]^2$). En la última columna se calcula la ganancia por el uso de *metatiling*, medida como la aceleración media en la devolución de una tesela frente a no utilizar *metatiling*, tal y como se refleja en la expresión (8.3).

$$G_{metatiling}(B) = \frac{\tau_{m,metatile\_n}(0)}{\tau_{m,metatile\_n}(B)} \tag{8.3}$$

Los resultados reflejan que la latencia involucrada en la petición de un *metatile* aumenta al hacerlo el tamaño del *buffer* utilizado. Sin embargo, aumenta en menor proporción que el número de teselas de que se compone el *metatile*. Por tanto, la latencia media para la obtención de cada tesela individual decrece al aumentar el tamaño del *metatile* solicitado al servidor de mapas remoto.

Un factor limitante a la hora de decidir el tamaño de metatile a utilizar es la cantidad de memoria que requiere la generación de la imagen. Nótese que para un factor de metatile de 5x5 (*buffer* de dos unidades), para una petición de tesela de 256x256 pixels la imagen generada es de 1280x1280 pixels.

Considérese, por ejemplo, que en el servidor de mapas GeoServer[1] (Deoliveira, 2008), la cantidad máxima de memoria permitida para una única petición WMS (parámetro *maxRequestMemory*) viene configurada por defecto a 64MB. Esta cantidad de memoria permite generar una imagen de 2048x2048 pixels con 4 bytes/pixel, lo que es equivalente a un *metatile* de 8x8. Si el SLD (*Styled Layer Descriptor*) contiene dos elementos *FeatureTypeStyle* para el dibujado de distintos tipos de línea, el tamaño máximo de la imagen se limita a 1448x1448 pixels (*metatile* máximo de 5x5).

Así, al realizar una petición de mapa de tamaño 5600×5600 píxels al servicio WMS de Cartociudad, el servidor GeoServer devuelve una imagen transparente de este mismo tamaño indicando el siguiente mensaje de error: «*Rendering request would use 91875KB, whilst the maximum memory allowed is 65536KB*».

8.1. Rendimiento del metatiling en la tarea de seeding

Como se comentó en el Capítulo 7, las técnicas de *seeding* permiten realizar una precarga de teselas, anticipándose a las peticiones de los usuarios, con el objetivo de mejorar la QoS. Una práctica habitual consiste en ir recorriendo en *zig-zag* la región geográfica para la generación de teselas. En este caso, es sencillo analizar las mejoras que pueden conseguirse mediante la aplicación del *metatiling*.

Suponiendo una rejilla rectangular de $M \times N$ teselas, para pregenerar todo el contenido mediante la técnica de *metatile* con tamaño de *buffer* B y centrada en la tesela solicitada (ver Figura 8.5), se requieren aproximadamente $\frac{M \times N}{(B+1)^2}$ peticiones al servidor de mapas remoto. Como se puede observar en dicha figura, el último *metatile* pedido contiene 16 teselas que ya estaban previamente en la caché, con lo que tan sólo un 36 % de la imagen aporta información nueva al sistema.

Mediante la estrategia de mínima correlación (ver Figura 8.6), el número de peticiones se reduce a $\frac{M \times N}{(2B+1)^2}$ aproximadamente, obteniendo una ganancia de $\frac{(B+1)^2}{(2B+1)^2}$. En la figura se muestra cómo los *metatiles* se desplazan de forma que no se producen solapamientos con aquellos previamente solicitados, maximizándose por tanto la información añadida al sistema.

Así, utilizando un tamaño de *buffer* de 3 unidades ($B = 3$), se reduce en un factor de 16 el número de *metatiles* solicitados al *backend*.

[1] http://geoserver.org

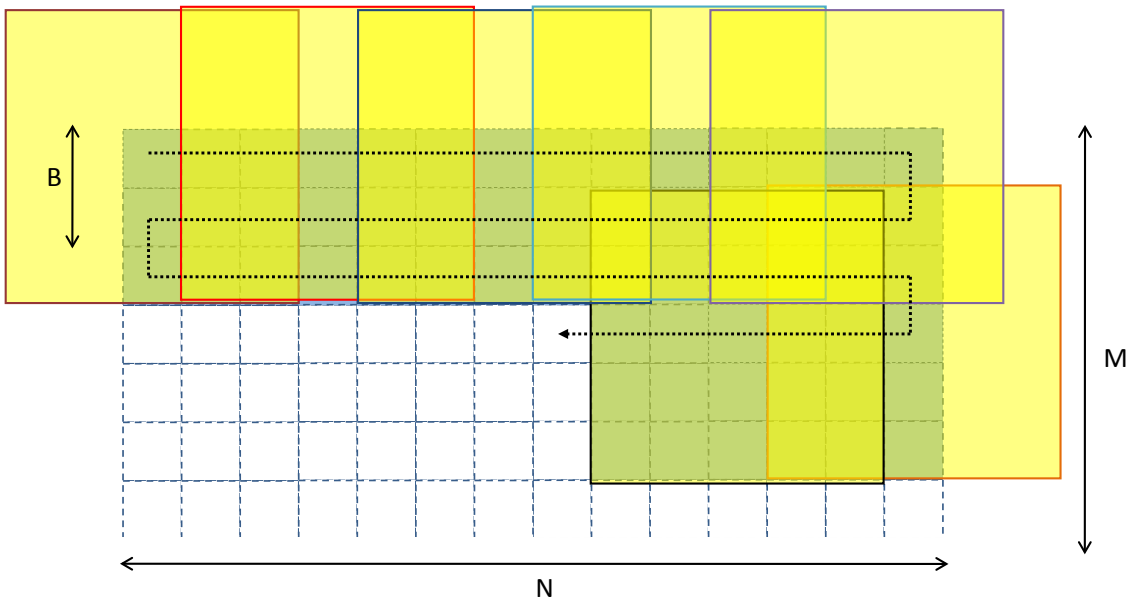


Figura 8.5: *Metatiles* pedidos al servidor de mapas remoto durante una tarea de *seeding*, utilizando *metatiles* centradas en la tesela a cachear, con $B = 2$.

Este mismo resultado podría obtenerse mediante el mecanismo de *metatiling* simple, solicitando las teselas con los desplazamientos oportunos. Sin embargo, el planificador de la tarea de precarga puede estar completamente desacoplado del *proxy*que implementa la técnica de *metatiling*. En este último caso, la estrategia de mínimima correlación propuesta permite optimizar la tarea de precarga.

8.2. Rendimiento del metatiling durante la carga dinámica

Para evaluar el rendimiento de las estrategias de *metatiling* durante el proceso de carga ante las peticiones de los usuarios, se ha utilizado de nuevo el prototipo de caché de teselas *WMSCWrapper*, descrito en el Apéndice A.2, como banco de pruebas.

Para simular las peticiones de los usuarios, se ha hecho uso de los registros de acceso o *logs* del servicio de mapas WMS-C de Cartociudad, estudiados en el Capítulo 5. De esta forma se consigue un patrón de acceso que refleja fielmente el comportamiento real de los usuarios, y los resultados son más representativos que en el caso de utilizar un patrón de peticiones "sintético".

Como servidor de mapas remoto se ha utilizado el servicio de CORINE. No se ha podido utilizar el propio servicio de Cartociudad, cuyos registros de peticiones se han analizado, por problemas en la disponibilidad del servicio.

Un total de 1.000.000 peticiones se han realizado a la caché, utilizando distintas configuraciones de *metatiling*. Para cada una de estas configuraciones, se ha recogido la tasa de aciertos de caché conseguida, y el número de objetos almacenados en la misma una vez completada la tarea. En todas las pruebas se parte de una caché vacía. Los resultados de los experimentos se muestran en las Figuras 8.8 y 8.7.

Como se puede observar, tanto la tasa de acierto como el número de objetos *cacheados*

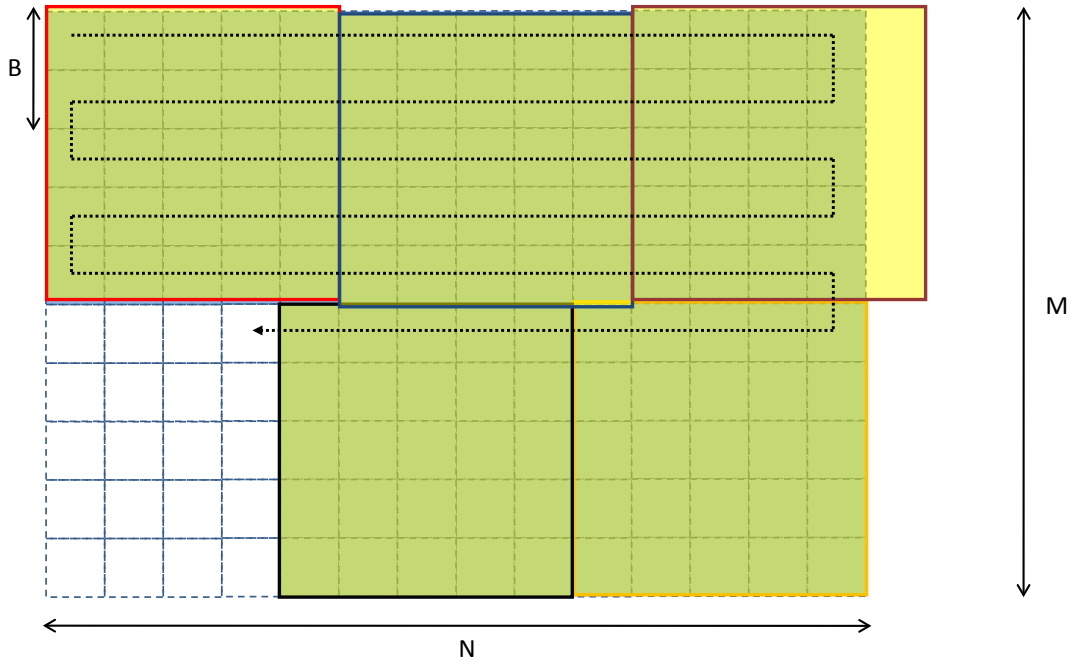


Figura 8.6: *Metatiles* pedidos al servidor de mapas remoto durante una tarea de *seeding*, utilizando la estrategia de *metatiling* de mínima correlación con la caché, con $B = 2$.

aumentan al hacerlo el tamaño del *buffer* utilizado. A igualdad de éste, ambas estrategias de *metatiling* obtienen unas tasas de acierto muy próximas. Sin embargo, mediante la configuración de mínima correlación, el número de objetos introducidos en la caché es notablemente superior que con la configuración de *metatile* centrado en la tesela solicitada. Esta diferencia aumenta al hacerlo el tamaño del *metatile*.

La ventaja conseguida con la configuración de *metatiling* de mínima correlación es que, manteniendo la tasa de fallos de caché, y por ello manteniendo el número de peticiones al servidor WMS remoto, se consigue una mayor cantidad de imágenes de mapa pregeneradas almacenadas en la caché para satisfacer más rápidamente posibles peticiones futuras.

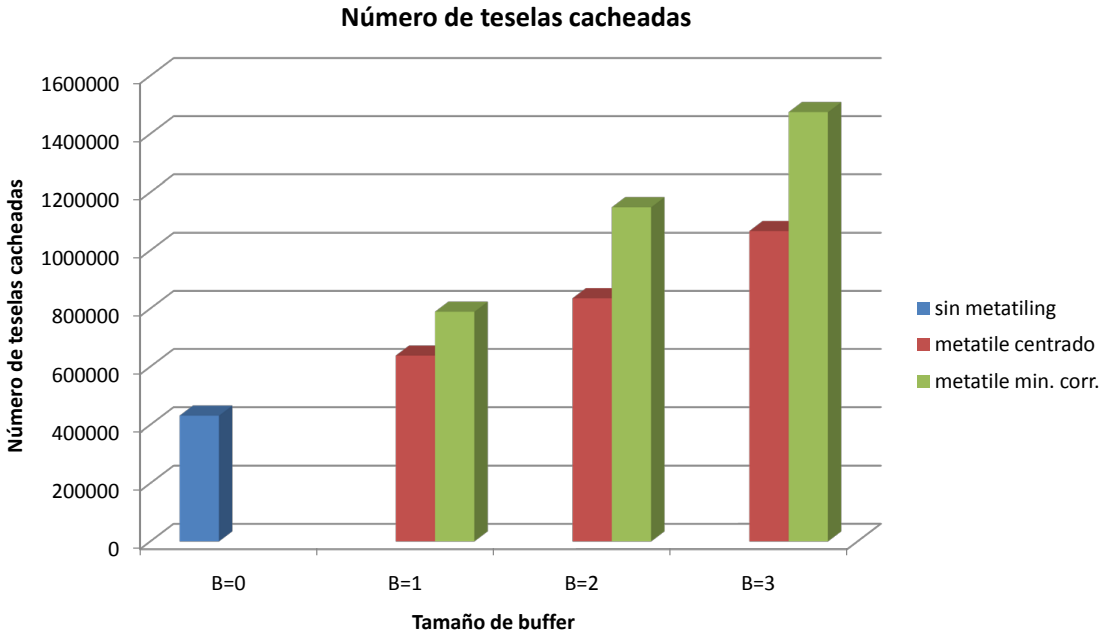


Figura 8.7: Comparativa de objetos cacheados al finalizar la tarea, para distintos tamaños de *buffer* y configuraciones de *metatile*, tras 1.000.000 de peticiones realizadas a la caché.

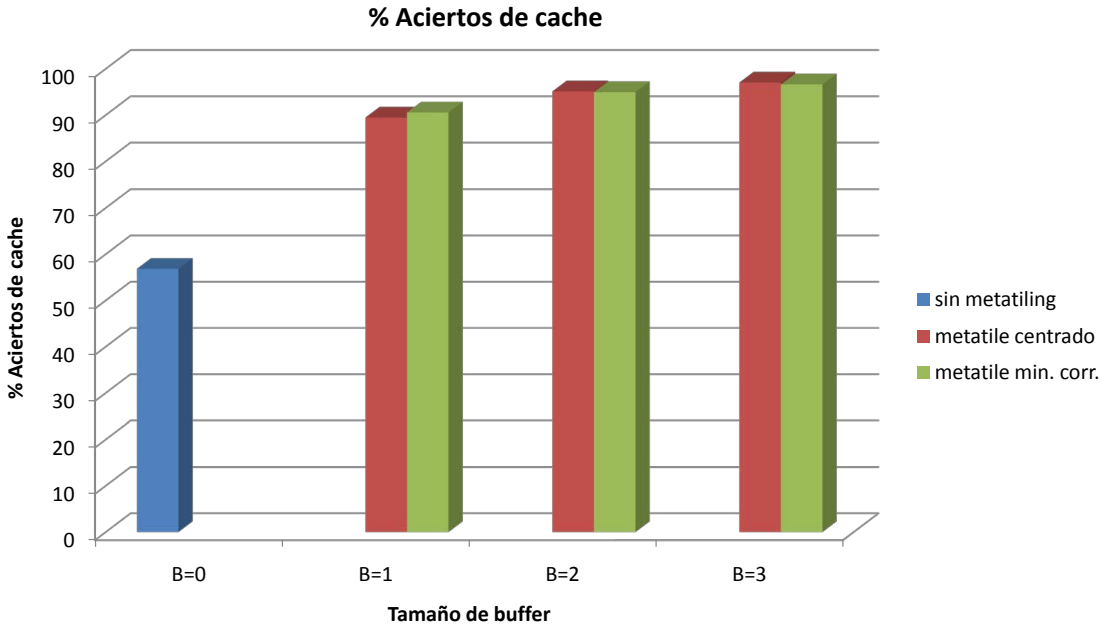


Figura 8.8: Comparativa de aciertos de caché, para distintos tamaños de *buffer* y configuraciones de *metatile*, tras 1.000.000 de peticiones realizadas a la caché.

Capítulo 9

Conclusiones y trabajos futuros

Resumen: En este capítulo se presentan las conclusiones obtenidas durante la realización de este trabajo de investigación y se ofrecen unas orientaciones que pueden servir como guía para continuar en un futuro con esta misma línea de investigación y desarrollo.

9.1. Conclusiones

Durante los últimos años se ha experimentado una gran proliferación en el uso de servicios SIG ofrecidos a través de la Web. Esto ha motivado la necesidad de disponer de servicios cada vez más escalables en las IDE para satisfacer esta incipiente demanda.

En el ámbito de los servicios de mapas, esto se tradujo en la aparición de nuevas especificaciones basadas en modelos teselados, como la recomendación WMS-C de OSGeo, el estándar WMTS de OGC, y otras soluciones no-estándar.

Los servicios de mapas teselados posibilitan la introducción de estrategias de caché mediante la discretización de los dominios de los parámetros de las peticiones. Esto permite distribuir imágenes pregeneradas a partir de una caché, la cuál puede situarse tanto en el lado del cliente, del servidor, o bien mediante una alternativa distribuida tipo P2P. De esta forma se mejora el tiempo de respuesta y la escalabilidad.

Sin embargo, en escenarios con recursos de almacenamiento escasos o en los que la cartografía se actualiza con mucha frecuencia, no resulta adecuado el pregenerar todo el contenido cartográfico. En estos casos generalmente se recurre al uso de cachés parciales que contienen tan solo un subconjunto de la cartografía.

Para garantizar una QoS aceptable es necesaria la actuación de adecuadas políticas de mantenimiento y gestión de estas cachés de mapas: estrategias de población inicial o *seeding*, mecanismos de carga dinámica ante las peticiones de los usuarios y políticas de reemplazo de caché.

A pesar de que estas estrategias actúan en distintas etapas de acuerdo con el estado de la caché, todas ellas comparten el mismo principio de funcionamiento; todas intentan mantener en caché los objetos con mayor probabilidad de ser pedidos. Este cacheo selectivo implica asumir que ciertas regiones son menos importantes que otras.

Durante el estudio del estado de arte llevado a cabo en esta tesis se ha comprobado que existe un reducido número de estas estrategias que sean específicas para los servicios

de mapas. La mayor parte de estrategias aplicadas a estos servicios son extraídas de otros ámbitos, como los proxies Web tradicionales, las cuáles no tienen en cuenta la componente espacial de los objetos que gestionan. En esta investigación se aborda la creación de nuevos algoritmos específicos para este dominio de aplicación que permitan optimizar el rendimiento de los servicios de mapas.

De esta forma, en esta tesis se han desarrollado puntos de mejora que abarcan el ciclo completo de un sistema de caché; ciclo que consiste en los siguientes pasos:

1. Durante la puesta en marcha del servicio la caché se encuentra inicialmente vacía, por lo que no se experimenta ninguna mejora por el uso de caché. Para mejorar la QoS experimentada por los usuarios, un mecanismo automático de precarga o *seeding* pobla parcialmente la caché con los objetos que se espera vayan a ser más pedidos.

2. Las peticiones de los usuarios que no han sido almacenadas durante el procedimiento anterior son cacheadas bajo demanda a medida que son pedidas por los usuarios para satisfacer más rápidamente posibles peticiones futuras.

3. Ante una petición de un usuario, un mecanismo de carga dinámica o *prefetching* puede predecir cuál podría ser la siguiente petición o grupo de peticiones y precargar este contenido de antemano para mejorar la experiencia del usuario.

4. Cuando la caché se encuentra completa un algoritmo de reemplazo debe determinar qué teselas deben ser reemplazadas para dejar espacio a nuevos objetos.

Para la precarga inicial o *seeding* de la caché se han propuesto dos tipos de modelos cuya filosofía es sustancialmente distinta:

- Una estrategia para la determinación de zonas prioritarias de caché mediante la aplicación de un modelo descriptivo. Estos modelos se basan en el uso de los registros de acceso a los servidores de mapas para determinar cuáles son las regiones de mapa más solicitadas por los usuarios. Los modelos descriptivos asumen un comportamiento estacionario en los patrones de acceso del servicio. Las altas tasas de acierto conseguidas experimentalmente validan la hipótesis de que puede realizarse una predicción del acceso futuro a las teselas atendiendo exclusivamente a la información disponible en accesos pasados. El análisis multi-resolución realizado respalda la utilización de estadísticas recogidas en un cierto nivel de la pirámide de escalas para predecir el comportamiento en otros niveles cercanos.

- A partir de los registros de uso de los servidores de mapas pueden extraerse fenómenos geográficos directores de las peticiones de los usuarios. Así, a partir de los registros de acceso de los servicios analizados en esta investigación, se aprecia que las zonas costeras, núcleos urbanos y las principales vías de transporte responden a este patrón. Se ha propuesto una arquitectura genérica que trata de anticipar posibles peticiones futuras a partir de un catálogo no-restringido de fenómenos geográficos y un breve histórico del servicio con peticiones pasadas. A partir de esta arquitectura genérica, se han propuesto sendos modelos predictivos. El primero de ellos emplea un estimador OLS para la parametrización automática del modelo. La otra propuesta utiliza un sistema inteligente basado en el uso de redes neuronales para asignar automáticamente prioridades a las teselas. Ambos modelos han sido validados experimentalmente obteniendo elevadas tasas de acierto en términos tanto de *Cache-Hit Ratio* como de *Byte-Hit Ratio*.

Habitualmente, los usuarios se desplazan de forma continua por el mapa, por lo que es probable que al recibir la petición de una tesela de mapa, las teselas adyacentes sean solicitadas a continuación. Aprovechando este hecho, se ha presentado una novedosa estrategia de carga dinámica basada en un *metatiling* adaptativo. Las implementaciones investigadas permiten únicamente la posibilidad de solicitar un *metatile* centrado en la tesela pedida por el usuario. Sin embargo, en un escenario de caché no completa (pero no vacía) esta elección del área a generar no resulta eficiente, puesto que es probable que algunas de las teselas próximas a la solicitada ya estén disponibles en la caché. Por ello, partiendo de la suposición de que la zona que engloba la tesela solicitada no está homogéneamente cargada, se ha desarrollado un algoritmo para la elección óptima de las *metatiles* a generar. El procedimiento propuesto busca obtener, en función del estado de la caché, la *metatile* que, conteniendo la tesela solicitada (no necesariamente centrado en la misma), complemente en mayor medida a la información almacenada en la caché. Se ha comprobado experimentalmente que, con la configuración de *metatiling* de mínima correlación propuesta, a igualdad de peticiones al servidor WMS remoto, se consigue una mayor cantidad de imágenes de mapa pregeneradas almacenadas en la caché para satisfacer más rápidamente posibles peticiones futuras.

Otra contribución de la tesis es la propuesta de dos nuevas estrategias de reemplazo de caché específicas para servicios de mapas:

- Una estrategia adaptativa basada en el uso de redes neuronales. Como entradas de la red neuronal se utilizan tres propiedades de las peticiones Web: actualidad de referencia, frecuencia de referencia, y el tamaño de la tesela referenciada, definidas todas ellas en torno a una ventana deslizante. Durante el proceso de entrenamiento, a cada registro correspondiente a la petición de una determinada tesela se le asocia una salida *booleana* (0 o 1) que indica si dicha tesela ha sido o no solicitada de nuevo dentro de la ventana. Una vez entrenada, la salida de la red neuronal será un número real en el rango [0,1] que debe interpretarse como la probabilidad de recibir una petición posterior del mismo objeto dentro de la ventana temporal. Con este modelo se han conseguido elevadas tasas de acierto en las predicciones realizadas sobre la *cacheabilidad* de las peticiones.

- Estrategia *Spatial-LFU*, variante de la política de reemplazo *Perfect-LFU* que aprovecha el principio de localidad espacial y la autocorrelación espacial entre las peticiones para reducir la sobrecarga derivada de almacenar las estadísticas de teselas individuales. Para evaluar el rendimiento de esta estrategia se ha desarrollado un simulador de caché de teselas que nos ha permitido realizar experimentos utilizando los registros de peticiones del servicio WMS-C de Cartociudad. Los resultados avalan que puede conseguirse una reducción un ahorro significativo en la sobrecarga de estas estrategias y al mismo tiempo mantener la tasa de acierto de caché a un nivel aceptable.

Para la experimentación y validación de las distintas políticas de gestión de caché propuestas, se han desarrollado diversas implementaciones prácticas de caché, liberadas ahora como software *Open Source* para su libre uso por la comunidad.

9.2. Futuras Líneas de Investigación

Durante la investigación presentada se han abierto nuevas vías de trabajo que todavía no se han abordado, las cuales se describen a continuación a modo de líneas futuras:

1. Las distintas estrategias de gestión de caché presentadas tienen como objetivo principal alcanzar cuanto antes una QoS objetivo y mantenerla durante el tiempo de vida de la caché. Por tanto, una importante línea de investigación es la formalización y definición de la calidad de servicio dependiente del tiempo para un sistema de caché no completo y su relación con la densidad de probabilidad por niveles $f_{req}(i, j, n)$ y en profundidad (ver Ecuación (3.8), pág. 28).

2. Encadenamiento de servicios de mapas teselados.

   a) Generación de metadatos automáticos a partir de los metadatos primarios y de la información añadida en la estructura de la caché.

   b) Labor de estandarización de QoS no contemplada por OGC. Inclusión en los metadatos de la información de calidad de servicio con el detalle que pueda obtenerse de los modelos de gestión (puede ser razonable incluir una segmentación horaria de la QoS en función de la carga observada en el pasado reciente). La QoS se podría definir según diferentes perspectivas temporales:

      - *Instantánea*: indicando la situación actual del servicio.
      - *Nominal*: indicando el valor asintótico ideal.
      - *Predictiva*: indicando mediante un modelo de evolución la QoS que se espera para un periodo de tiempo a medio plazo. Este parámetro podría caracterizar bien un sistema *proxy cache* que esté en fase transitoria de carga de su caché (P.ej. iniciando el servicio desde un estado vacío o realizando una actualización masiva de información).

   c) Composición en cascada de la calidad de servicio en función de las fuentes primarias de teselas.

3. Adaptación de los algoritmos propuestos para su aplicación a otros servicios OGC con carácter general. Los servicios WFS y WCS podrían ser buenos candidatos.

4. Identificación de usuarios individuales para optimizar los algoritmos de precarga dinámica. Esto permitiría extraer pautas de comportamiento a partir de muestras menos ruidosas. Sin embargo, la existencia de *firewalls* corporativos y servidores *proxy* complica esta tarea. Pueden emplearse heurísticas para ayudar en la identificación de usuarios individuales. Incluso si la dirección IP es la misma entre peticiones, si el *UserAgent* refleja un cambio en el software utilizado para realizar la navegación, o en el Sistema Operativo, se puede realizar una suposición razonable de que cada tipo de *UserAgent* para una IP dada representa a diferentes usuarios (Cooley et al., 1999).

5. Experimentación con otros métodos regresivos en el modelo predictivo. Por ejemplo, una regresión WLS (*Weighted Least Squares*) podría reducir la influencia de las teselas no-pedidas en el modelo final.

6. Adaptación del factor de simplificación utilizado por los algoritmos de gestión en función de la auto-correlación medida en los registros acceso del servicio, para cada nivel de resolución (hay que destacar que la auto-correlación espacial es altamente dependiente de la escala) y por zonas. Otra aproximación prometedora podría ser el uso de un modelo local, p.ej. con una regresión GWR (*Geographically Weighted Regression*). Esto reflejaría que la influencia de algunos fenómenos en la atención del

usuario varía dependiendo de la localización geográfica. Se propone explorar el uso de técnicas de detección de *clusters* para crear subdominios y aplicar las estrategias de gestión de caché por zonas.

7. Perfeccionamiento del selector de fenómenos utilizado en el modelo predictivo para la precarga de teselas. Con el objetivo de mantener el modelo lo más sencillo posible y reducir su carga computacional, se propone la aplicación de técnicas de reducción de datos.Se propone incorporar el análisis "semántico" de los fenómenos de interés para enriquecer el catálogo incorporando fenómenos relacionados a través de servicios de búsqueda mediante metadatos.

8. Transformación del modelo predictivo para que sea adaptativo. En el caso del modelo de precarga basado en redes neuronales esto se conseguiría de forma sencilla sustituyendo el algoritmo de entrenamiento por lotes (*batch*) actual, por un entrenamiento incremental (*on-line*).

Apéndice A

Prototipos de caché de teselas

Resumen: Durante el transcurso de esta investigación se han desarrollado diversos prototipos de caché. Estos prototipos, implementados originalmente para experimentar con los distintos algoritmos de gestión de una caché, han sido liberados como software *Open Source* para su libre uso por la comunidad. El prototipo WMSCWrapper desarrollado es una implementación completa y funcional de una caché de teselas programada en Java. Dada la popularidad de GeoWebCache, algunas de las estrategias propuestas en esta tesis, como el modelo predictivo para la precarga inicial presentado en el Capítulo 7.3 y el mecanismo de carga dinámica basado en *metatiling* adaptativo presentado en el Capítulo 8, han sido implementadas sobre esta herramienta. Por último, se ha implementado un simulador de caché para la evaluación del rendimiento de múltiples estrategias de reemplazo de caché, que incluye tanto algunas de las estrategias más populares encontradas en la literatura como las propuestas aquí en el Capítulo 6.

A.1. Comparativa de Implementaciones de Caché

La estandarización de los servicios de mapas teselados ha motivado la aparición de múltiples implementaciones de cachés de teselas. Entre ellas destacan: TileCache, GeoWebCache y MapProxy. En la Tabla A.1 se muestra una comparativa entre las implementaciones anteriores.

Como se puede observar, tanto TileCache como MapProxy están implementados en Python, mientras que GeoWebCache está desarrollado en Java. Todos ellos ofrecen interfaces WMS-C, TMS y KML. GeoWebCache y MapProxy también ofrecen el servicio WMTS de OGC. Además, GeoWebCache puede recombinar y remuestrear teselas para responder peticiones WMS arbitrarias, y también puede utilizarse para servir mapas a los clientes Google Maps y Microsoft Bing Maps.

En cuanto al almacenamiento de las teselas, todas las implementaciones ofrecen la posibilidad de albergar las imágenes de mapa directamente en el sistema de ficheros. TileCache y GeoWebCache soportan también la especificación MBTiles[1] para el almacenamiento de teselas de mapa en una base de datos SQLite, para su uso inmediato o para su transferencia. MapProxy soporta Apache CouchDB[2] como almacén de teselas. Se trata de un gestor de

[1] http://mapbox.com/mbtiles-spec/
[2] http://couchdb.apache.org/

	TileCache	***GeoWebCache***	***MapProxy***
Compañía	Metacarta Labs	OSGeo	Omniscale
Lenguage	Python	Java	Python
Servicios soportados	WMS-C, TMS, KML	WMS, WMS-C, TMS, WMTS, KML, Google Maps, Bing Maps	WMS, WMS-C, TMS, WMTS, KML
Almacenamiento teselas	Disk, GoogleDisk, Memcached, Amazon S3, MBTiles	Disk	Disk, MBTiles, CouchDB
Almacenamiento metadatos	No	Sí	Sí
Políticas reemplazo	LRU	LRU, LFU	None
Selección de zonas de caché	bounding box círculo	bounding box	bounding box Geometrías WKT y fuentes OGR
Meta Tiles	Sí	Sí	Sí
Meta Buffer	Sí	Sí	Sí
Reproyección al vuelo	No	Sí (con Geoserver)	Sí (nativo)

Tabla A.1: Resumen de características de las principales implementaciones de caché de teselas de código abierto.

bases de datos orientado a documentos que puede ser indexada y pueden realizarse consultas según la filosofía de MapReduce. Por su parte, TileCache permite almacenar las teselas de mapa en la nube mediante el servicio S3 de Amazon[3], o mantenerlas en memoria utilizando Memcached[4].

GeoWebCache mantiene los metadatos asociados a las teselas, como el tiempo de último acceso o el número de veces que se ha pedido cada tesela. Esto le permite utilizar políticas de reemplazo de caché como LRU y LFU. TileCache soporta LRU usando el tiempo de último acceso recogido por el sistema operativo.

Estos servicios permiten especificar una zona geográfica para la actuación de ciertos mecanismos de gestión de la caché: precarga, actualización y borrado. Por ejemplo, en TileCache puede especificarse una zona de precarga definida por un cierto *bounding* rectangular o un círculo con centro y radio dados. GeoWebCache tan sólo soporta zonas rectangulares. En este sentido MapProxy es la más completa, permitiendo tres métodos para describir la zona de interés: un *bounding box* rectangular, un fichero de texto con una o más geometrías en formato WKT, o geometrías obtenidas a partir de cualquier fuente de datos OGR (p.ej. Shapefiles, PostGIS, etc.).

[3] http://aws.amazon.com/es/s3/
[4] http://memcached.org/

Los tres servicios soportan tanto *metatiling* como *meta-buffer*. El *meta-buffer* consiste en añadir un *buffer* extra a los bordes de la zona solicitada, para reducir el efecto del etiquetado redundante.

Cuando se recibe una petición de una tesela en un CRS no soportado, GeoWebCache y MapProxy permiten la re-proyección al vuelo de teselas en una proyección disponible. GeoWebCache lo hace a través de GeoServer, mientras que MapProxy lo permite de forma nativa.

A.2. Prototipo de caché de teselas WMSCWrapper

> Los contenidos del presente apartado han sido parcialmente publicados en (García et al., 2011b; García y de Castro, 2010b,c).

Para la experimentación y validación de muchos de los algoritmos propuestos en esta investigación, se ha desarrollado un prototipo de caché de teselas. Esta implementación, bautizada como WMSCWrapper, está disponible como proyecto *Open Source* y tiene una arquitectura adecuada para la inclusión de componentes y sondas experimentales. Otro motivo por el que se optó por desarrollar un nuevo sistema de caché fue que el resto de implementaciones de filtros WMS-C examinadas existentes están fuertemente orientadas hacia el exclusivo soporte del servicio teselado de mapas para obtener la mencionada ganancia de caché.

Esta implementación, elegida como banco de pruebas para los experimentos, es un *proxy web* que se enmarca, dentro del diagrama taxonómico de Shan y Hua (2009) (ver Figura 3.1, pág. 24), en la *Web Layer* como un *Web caching*, y en la taxonomía vertical como una *Object Cache*, en la que además se implementan funcionalidades de procesado típicas de la capa de aplicación.

WMSCWrapper es un servicio implementado en Java, siguiendo la especificación J2EE, como un conjunto de *servlets* que exponen los métodos de la recomendación WMS y la extensión WMS-C en una interfaz OGC y permiten también el acceso a la información *cacheada* por medio de otras interfaces como los métodos REST utilizados por *Google Earth* al solicitar teselas en KML (*Keyhole Markup Language*).

Se configura de forma bastante simple mediante la especificación de una serie de capas disponibles en otro servicio WMS y unas características de tratamiento de los objetos en la caché. Cada petición es analizada en busca de los parámetros obligatorios y opcionales de la recomendación y después transferida a una serie de componentes intercambiables que pueden preprocesar o prostprocesar la información según las necesidades.

Existen componentes que realizan las siguientes operaciones:

- Comprobación de la validez de la petición.
- Obtención de teselas del servicio del *backend*.
- Ejecución de algoritmos de *metatiling* con procesamiento multi-hilo.
- Inclusión de marcas de agua.
- Etiquetado para depuración y supervisión del funcionamiento del servicio.

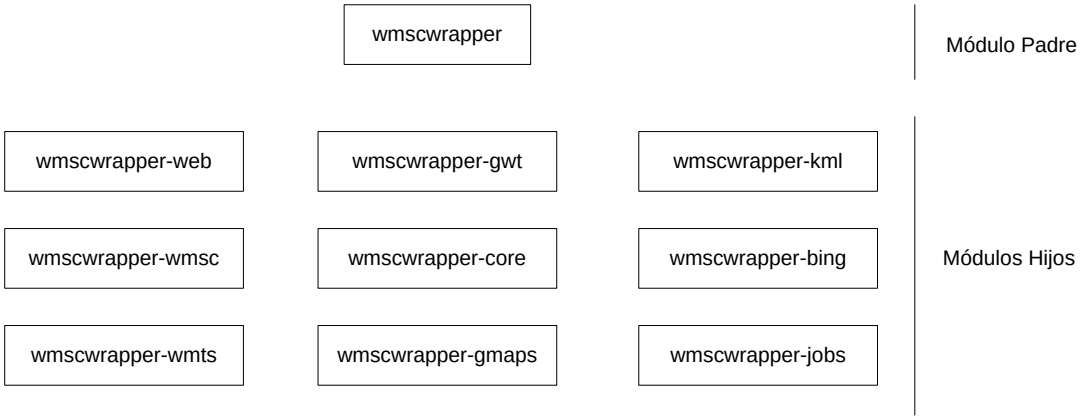


Figura A.1: Configuración multi-módulo del proyecto Maven de WMSCWrapper.

- Mantenimiento de la caché.
- Recopilación de estadísticas.

A.2.1. Arquitectura

La aplicación ha sido desarrollada utilizando la herramienta de software para la gestión de proyectos Maven de Apache[5]. Esta herramienta facilita la tarea de desarrollo en todas sus fases, desde el proceso de compilación, descarga automática de las dependencias, despliegue de la aplicación o incluso documentación de la misma. Para mejorar la modularidad de la aplicación, ésta ha sido construida como un proyecto multi-módulo de Maven, con un módulo "padre" y varios "hijos" como se muestra en la Figura A.1.

De esta forma se aíslan los diferentes módulos para su posible reutilización en otros desarrollos, y se facilita la incorporación de nuevos módulos al sistema. Cada módulo recoge una funcionalidad concreta:

- *wmscwrapper-core*: núcleo del sistema de caché, utilizado por el resto de los módulos.
- *wmscwrapper-web*: módulo web que contiene el servlet principal de la aplicación.
- *wmscwrapper-wmsc*: interfaz compatible con la especificación WMS-C de OSGeo.
- *wmscwrapper-wmts*: interfaz compatible con el estándar WMTS de OGC.
- *wmscwrapper-gmaps*: interfaz para el cliente Google Maps.
- *wmscwrapper-bing*: interfaz para el cliente de Microsoft Bing Maps.
- *wmscwrapper-kml*: pasarela KML para Google Earth.
- *wmscwrapper-gwt*: interfaz de usuario AJAX desarrollada en Google Web Toolkit.
- *wmscwrapper-jobs*: tareas de mantenimiento de la caché.

Asimismo, se ha hecho uso del *framework* de código abierto Spring[6]. Este *framework* permite el desarrollo de componentes intercambiables, llamados "*beans*", que son orquestados mediante un fichero XML, pudiendo intercambiar componentes sin necesidad de recompilar el proyecto.

[5] http://maven.apache.org/.
[6] http://www.springsource.org/

El *DispatcherServlet* actúa como punto de entrada de la aplicación, despachando las peticiones hacia la interfaz correspondiente mediante un *HandlerMapping*, que realiza un mapeo de la URL de la petición con el controlador que debe procesarla en función de la interfaz utilizada.

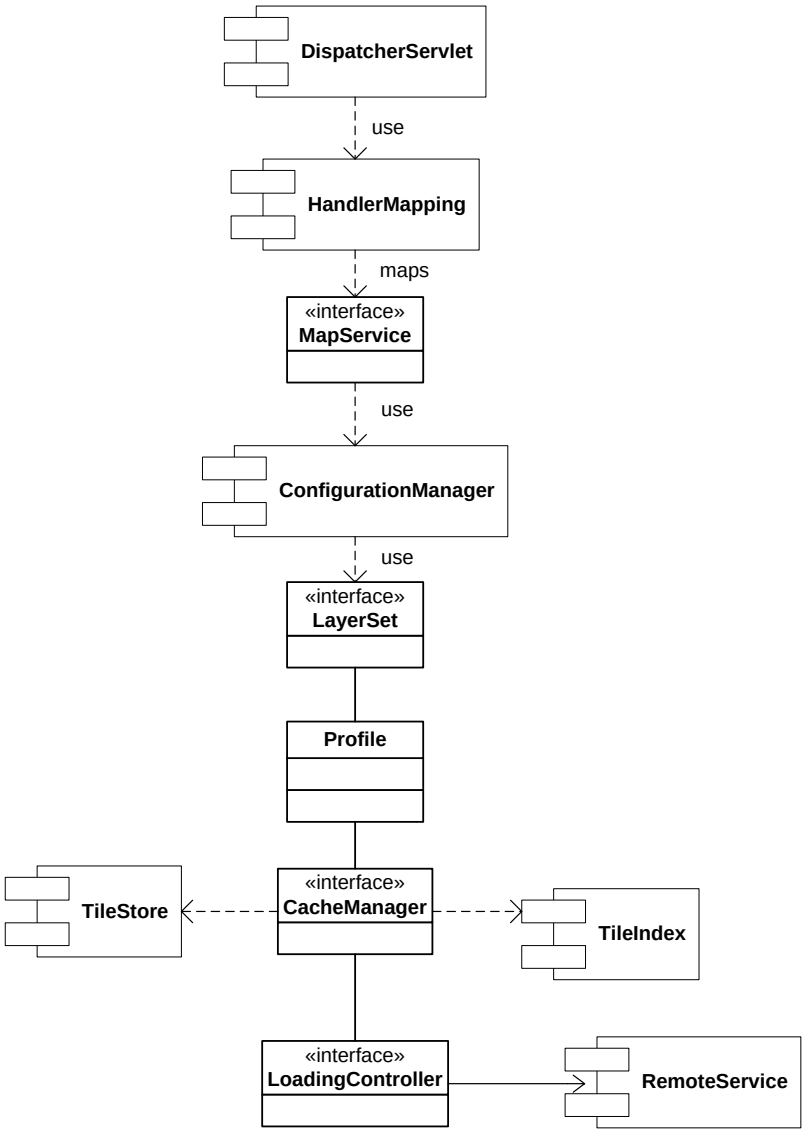


Figura A.2: Diagrama de componentes (simplificado) de la caché WMSCWrapper.

A.2.1.1. Interfaces de servicio

WMSCWrapper es capaz de ofrecer el servicio a través de múltiples interfaces de acceso: actualmente como WMS *proxy* según la especificación WMS-C, a través de una interfaz estándar WMTS, y otras especificaciones no estándar como Google Maps y Microsoft Bing Maps, ó actuando como pasarela KML para Google Earth.

WMS-C

El *proxy* caché desarrollado es compatible con la especificación WMS-C de OSGeo. Se han implementado las operaciones GetCapabilities y GetMap, para la obtención de los metadatos de servicio y la devolución de una imagen de mapa, respectivamente.

WMTS

Se han implementado las operaciones GetCapabilities y GetTile definidas en este estándar. Las peticiones pueden codificarse tanto en KVP como en REST (*Representational State Transfer*).

KML

El *proxy* caché implementado puede funcionar como pasarela KML para el cliente Google Earth. Al solicitar una capa en este formato, como respuesta se devuelve un documento KMZ (KML comprimido) interpretable por este cliente. El documento KML contiene un elemento <*NetworkLink*>, que contiene, a su vez, un elemento <*Link*> mediante el cual se solicita el contenido que se desea visualizar, indicado a través del elemento <*viewFormat*>. El contenido se actualiza periódicamente, o cada vez que se detiene la navegación. Al recibir las peticiones de los <*Link*> se devuelve un fichero KMZ generado dinámicamente, que contiene una matriz de elementos <*GroundOverlay*>, cada uno de los cuales contiene una única petición conforme al estándar WMS-C. Este mismo procedimiento puede consultarse en (Wernecke, 2008; Honda et al., 2006; Smith y Lakshmanan, 2006; McClendon et al., 2011).

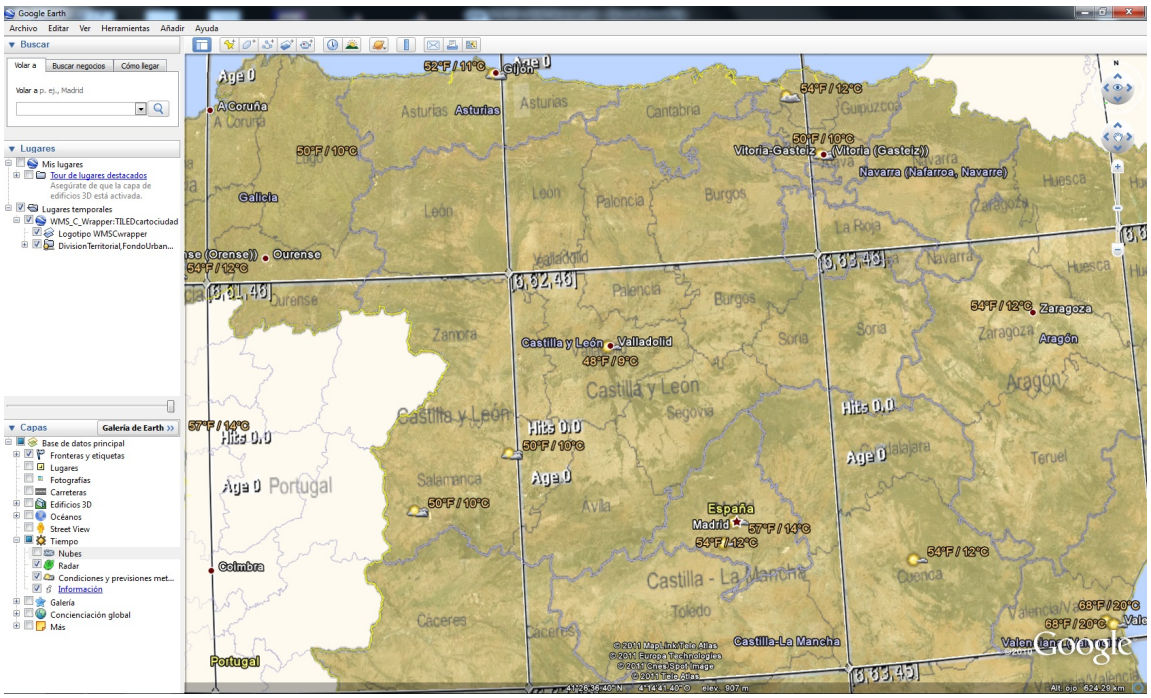


Figura A.3: Captura de Google Earth con cartografía obtenida de la caché WMSCWrapper en modo *debug*.

Google Maps

Google Maps utiliza una proyección esférica mercator, por lo que la capa a visualizar debe estar disponible en el sistema de coordenadas denominado EPSG:900913 (muy similar al EPSG:3857). En el Listado A.1 se muestra el código JavaScript necesario para superponer una capa procedente de la caché WMSCWrapper[7].

Figura A.4: Captura de Google Maps con cartografía obtenida de la caché WMSCWrapper en modo *debug*.

Listado A.1: Código JavaScript para superponer en Google Maps una capa accesible a través de WMSCWrapper

```
var tilelayer = new GTileLayer(<copyrights>, <minResolution>, <maxResolution>,
{
  tileUrlTemplate: 'http://<host>:<port>/WMS_C_wrapper/gmaps?layers=<layer_name>
      &zoom={Z}&x={X}&y={Y}&format=<format>',
  isPng:<{true/false}>,
  opacity:<opacity>
} );
```

Los índices X, Y, Z anteriores son sustituidos por el cliente de Google por los valores apropiados de coordenadas latitudinal, longitudinal, y nivel de zoom, respectivamente, según la localización a visualizar en el mapa.

El esquema de teselado utilizado por Google Maps tan sólo difiere del utilizado internamente por WMSCWrapper en la inversión del índice Y, por lo que la traducción es inmediata: $Y_{WMSCWrapper} = MaxY - Y_{GoogleMaps}$.

[7]Información adicional en la documentación del API de Google Maps: `http://code.google.com/intl/es-ES/apis/maps/documentation/reference.html#GTileLayer`

Bing Maps

Bing Maps utiliza la misma proyección que Google Maps por lo que se requiere que, al igual que para este último, la capa esté disponible en esta proyección.

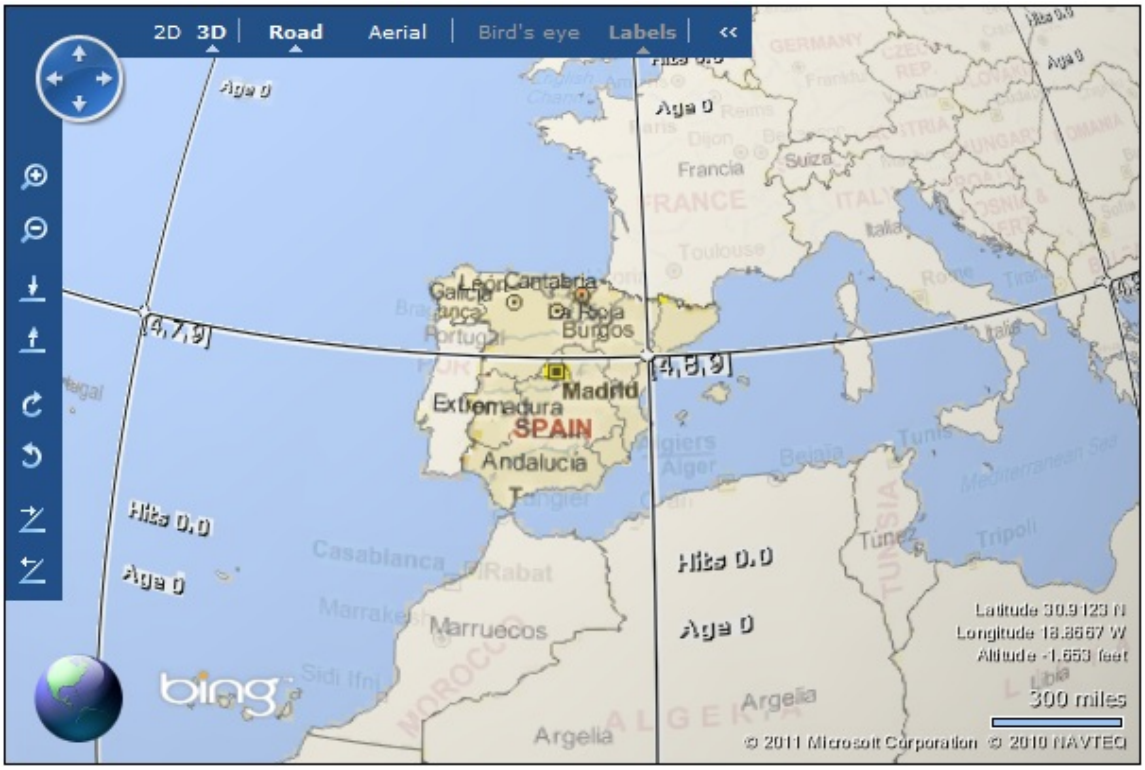


Figura A.5: Captura de Bing Maps con cartografía obtenida de la caché WMSCWrapper en modo *debug*.

Listado A.2: Código JavaScript para superponer en Bing Maps una capa accesible a través de WMSCWrapper.

```
var map = new VEMap('myMap');
var tileSourceSpec =
new VETileSourceSpecification('TITLE_OF_LAYER',
'http://<host>:<port>/WMS_C_wrapper/bing?quadkey=%4&format=image/png&layers=<
    layer_name>'
);

tileSourceSpec.Opacity = 0.5;
map.AddTileLayer(tileSourceSpec, true);

...

<body onload="GetMap();">
```

Para optimizar el indexado y almacenamiento de las teselas, las coordenadas XY de una tesela se combinan en una cadena uni-dimensional denominada *quadkey*, tal y como se describió en el Capítulo 2.2.3. Al recibir una petición procedente de Bing Maps, el *proxy* WMSCWrapper traduce el *quadkey* recibido en las coordenadas XY correspondientes.

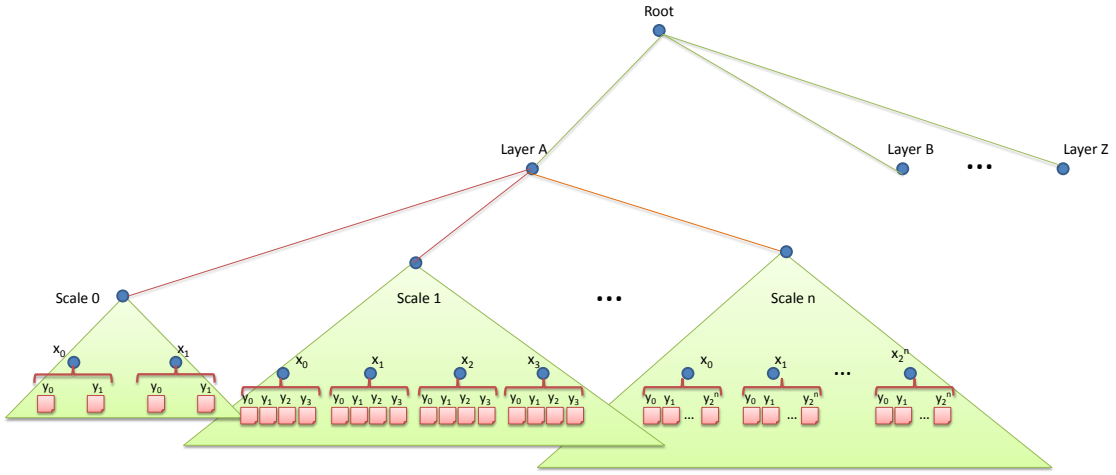


Figura A.6: Estructura de directorios para el almacenamiento persistente de las teselas en WMSCWrapper.

En el Listado A.2 se muestra el código JavaScript necesario para superponer en Bing Maps una capa accesible a través de WMSCWrapper. En la Figura A.5 se muestra una captura de este cliente mostrando cartografía obtenida a partir de la caché con el modo de depuración activado.

A.2.1.2. LayerSets

El servidor puede gestionar simultáneamente diversas cachés de teselas, denominadas *LayerSets*, con configuraciones inicialmente independientes. Estas *LayerSets* se pueden exportar a través de las distintas interfaces de acceso comentadas anteriormente. Además de la propia capa de mapa (*WMSLayerSet*), se ha añadido la posibilidad de visualizar otras capas asociadas a ésta. Estas capas son *CachedLayerSet* y *StatsLayerSet*. La primera de ellas permite ver sobre el mapa qué teselas están cacheadas y cuáles no (ver Figura A.15). La segunda representa en forma de *heatmap* o mapa de calor las estadísticas de las peticiones recibidas (ver Figura A.8).

A.2.1.3. Almacén de teselas (*TileStore*)

El sistema de almacenamiento de las teselas es muy flexible, permitiendo que éstas sean almacenadas tanto en disco como en múltiples Sistemas Gestores de Bases de Datos (SGBD).

El almacenamiento de las teselas en disco se realiza siguiendo la siguiente estructura de directorios: `raíz/layer/resolución/x/y.<extension>` (ver Figura A.6).

Sin embargo, a partir del módulo de persistencia implementado pueden extenderse, de forma sencilla, otros componentes que utilicen diferentes estructuras de directorios. Esto permite realizar una optimización en función del sistema de ficheros utilizado.

Asimismo, se han implementado diversos componentes para el almacenamiento de las teselas en distintos SGBD como MySQL, PostgreSQL, HSQLDB o Derby. El almacena-

miento de los objetos en base de datos permite aprovechar el indexado ofrecido por estas. En algunos escenarios, con un reducido número de teselas a cachear, puede hacerse uso de bases de datos en memoria (IMDB - *In-memory database*) como Derby o HSQLDB. Iniciativas como Oracle GeoRaster o PostgisRaster (WKTRaster) respaldan el uso de bases de datos para el almacenamiento de datos ráster.

A.2.1.4. Recopilación de estadísticas con índice espacial (*TileIndex*)

Se ha implementado un componente de caché espacial cuyo objetivo principal es llevar un registro espacial en tiempo real de la actividad de la caché. Esta actividad queda registrada en un índice *QuadTree* mediante el cual se pueden realizar búsquedas espaciales de gran eficacia, especialmente los recorridos en profundidad. Dada la naturaleza exponencial de la estructura de la pirámide, resulta impráctico albergar un índice del mismo tamaño que la caché que queremos representar, por lo que se ha implementado un *Quadtree* truncado en un nivel concreto que seleccionamos mediante un parámetro. Es evidente que ajustando este parámetro se puede transformar el índice para que cubra completamente la pirámide de teselas.

Este índice puede utilizarse para almacenar cualquier característica primaria o derivada que generen nuestros algoritmos o necesiten las heurísticas de nuestros experimentos.

Se ha implementado una estrategia básica inicial para obtener una aproximación a la probabilidad de acceso a las zonas geográficas representadas por las teselas.

Dado que no resulta práctico realizar operaciones a resoluciones altas, es preciso consolidar características espaciales en niveles intermedios de forma que esta información nos permita extrapolar y predecir las características de los niveles inferiores, que consideraremos inaccesibles. Se ha implementado el esquema ilustrado en la Figura 3.4 (pág. 30).

La probabilidad de acceso a una tesela $P_{req}\left\{T\left(i,j,n\right),t\right\}$ (ver expresión (3.1), pág. 26) es uno de los parámetros más importantes que caracterizan el comportamiento de los usuarios de la caché y puede estimarse estadísticamente. Partiendo de la condición previa de que las peticiones están restringidas a la rejilla de referencia, la probabilidad de acceso a una tesela de coordenadas $T\left(i,j,n\right)$ puede aproximarse en base a la historia de las peticiones pasadas mediante:

$$P_{req}\left\{T\left(i,j,n\right)\right\} = \lim_{N\to\infty} \frac{n_{req(i,j,n)}}{\sum\limits_{i,j,n} n_{req(i,j,n)}} \simeq \frac{n_{req(i,j,n)}}{N_{total}} \tag{A.1}$$

donde $n_{req(i,j,n)}$ es el número de veces que se ha pedido la tesela $T\left(i,j,n\right)$ y N_{total} es el número total de peticiones recibidas.

De acuerdo con (A.1) el índice espacial registra cada petición de elementos $T\left(i,j,n\right)$ mediante dos contadores de "*hits*" que reflejan la historia a medio y largo plazo de la zona geográfica cubierta por esa tesela. Cada *hit* en un determinado nivel se anota en las estadísticas de las teselas de niveles superiores implementando una aproximación a la expresión (3.8). Este comportamiento se ilustra en la Figura A.7. De esta forma la probabilidad de un *bounding box* geográfico se calcula como el cociente del número de peticiones englobadas y el total de peticiones del sistema.

Dado que el área geográfica cubierta por cada elemento $T\left(i,j,n\right)$ depende exponencialmente de la escala n, el número de peticiones recibidas aumenta a medida que n disminuye y la probabilidad de acceso deja de ser una buena heurística en los algoritmos que operen en distintas escalas. Por ese motivo las heurísticas implementadas utilizan, en lugar de la

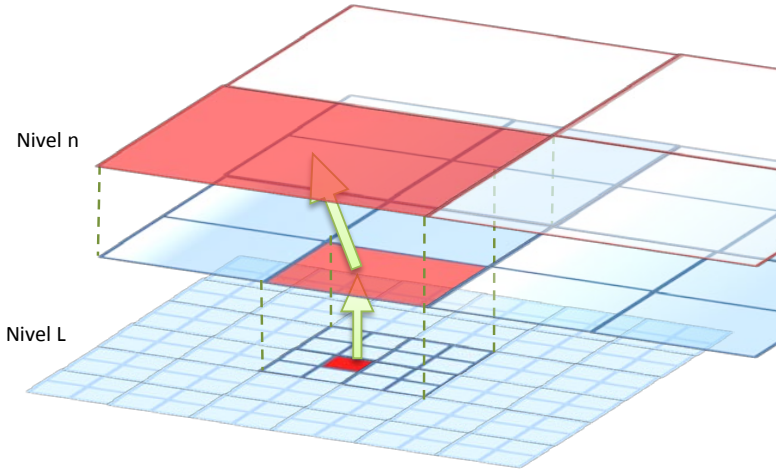


Figura A.7: Recopilación de estadísticas en el índice espacial

probabilidad, la función densidad de probabilidad media (3.1) de cada tesela. En esta implementación es inmediato interpretar dicha función como la densidad espacial normalizada de *hits* de caché:

$$P_{req}\left\{T\left(i,j,n\right)\right\} = f_{req}\left(x,y,n\right)\Delta x\Delta y = \frac{n_{h(i,j,n)}}{N_{total}}$$
$$f_{req}\left(x,y,n\right) = \frac{n_{h(i,j,n)}}{\Delta x\Delta y N_{total}} \tag{A.2}$$

Con este sistema de estimación de $f_{req}\left(x,y,n\right)$ se pueden implementar algoritmos de delimitación de zonas homogéneas (Z_i) haciendo equivalente (A.2) a la densidad espacial normalizada de peticiones. En la Figura A.8 se muestra el resultado de la actividad de este índice espacial en un nivel de la caché.

Este prototipo de caché permite utilizar índices de tipo *quadtree* que se cargan en memoria y se persisten periódicamente en disco. Por otra parte, incorpora la posibilidad de recoger las estadísticas de la actividad de la caché en una base de datos PostgreSQL con la extensión espacial PostGIS. Este tipo de índice resulta de gran utilidad para realizar un posterior análisis de las estadísticas almacenadas. Asimismo, se ha creado un tipo de índice *PonderatedIndex* que permite combinar la información de otros índices existentes de forma ponderada asociando un peso a cada uno de ellos.

A.2.1.5. LoadingController

Cuando la tesela solicitada no está almacenada en la caché (diremos que se ha producido un fallo de caché) ésta debe ser pedida al servidor de mapas remoto a través de una petición WMS estándar. Esta es la tarea del *LoadingController*. Se han implementado dos de estos componentes como se describe a continuación. En un futuro se pretende desarrollar módulos que permitan utilizar un *backend* que no sea un servidor WMS, como por ejemplo un servicio WMTS, y encadenar servicios en cascada.

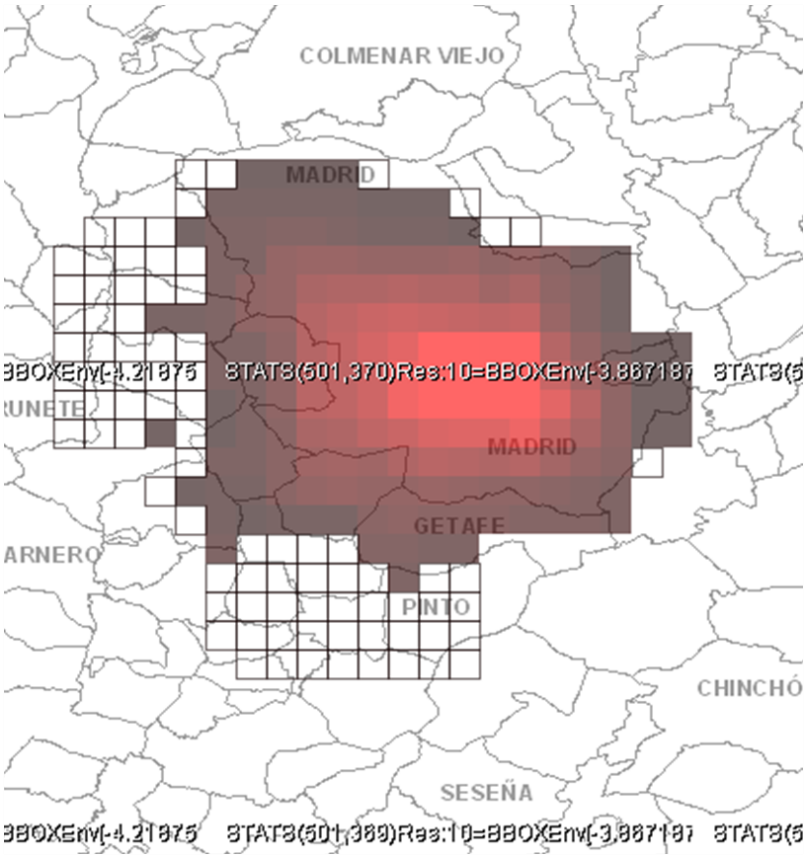


Figura A.8: Capa de teselas virtuales (*StatsLayerSet*) representando una de las estadísticas recopiladas junto con información de depuración.

BasicLoadingController

El componente *BasicLoadingController* realiza una petición WMS para obtener la imagen deseada a partir del servicio de mapas remoto. La URL de la petición se construye a partir de los parámetros de configuración del *LayerSet* y el *bounding box* correspondiente a la tesela solicitada.

BufferedLoadingController

El componente *BufferedLoadingController* utiliza la técnica de generación de peticiones de mayor tamaño que la tesela a cachear (*metatiling*) descrita en el Capítulo 8. Este componente puede configurarse para adoptar tanto la técnica de *metatiling* centrado en la tesela solicitada, como la estrategia de mínima correlación con la caché. Además, permite escoger el tamaño de *metatile* óptimo para cada petición, en base al estado de la caché y a la latencia media para la obtención de un *metatile* del servidor de mapas en función de su tamaño. La implementación llevada a cabo incluye además un procedimiento en segundo plano para realizar el postproceso de las teselas adicionales. De esta forma se busca minimizar también el tiempo necesario para obtener un objeto directamente de la caché.

A.2.2. Configuración del servicio

Toda la configuración de la caché se realiza a través ficheros de *beans* de Spring. En el Listado A.3 se muestra un ejemplo de configuración de una capa de mapa.

Listado A.3: Configuración de una capa de mapa en la caché.

```
<bean id="cartociudad_lyr" class="org.uva.idelab.wmscwrapper.layer.WMSLayerSet">
  <property name="name" value="cartociudad"/>
  <property name="description" value="cartociudad layer"/>
  <property name="url" value="http://www.cartociudad.es/wms/CARTOCIUDAD/
      CARTOCIUDAD?"/>
  <property name="layers" value="DivisionTerritorial,FondoUrbano,Vial,Portal,
      Toponimo,SeccionCensal,CodigoPostal"/>
  <property name="srs" value="EPSG:4326"/>
  <property name="extension" value="png"/>
  <property name="profile" ref="globalGeodetic20Levels"/>
  <property name="bbox" ref="world_envelope"/>
  <property name="cacheManager">
    <bean id="cartociudadCacheManager" parent="SpatialDiskCacheManager">
      <property name="tileStats">
        <bean name="fileIdx" class="org.uva.idelab.wmscwrapper.cache.index.
            model.dao.file.FileIndexDAO">
          <property name="path" value="${basePath}/cartociudad_stats" />
        </bean>
      </property>
      <property name="tileStore">
        <bean id="cartociudadDiskTileStore" class="org.uva.idelab.wmscwrapper.
            tilestore.DiskTileStore">
          <property name="basePath" value="${basePath}/cartociudad_store" />
        </bean>
      </property>
    </bean>
  </property>
  <property name="debugmode" value="true"/>
  <property name="controller">
    <bean id="bufferedLoadingController" class="org.uva.idelab.wmscwrapper.
        client.BufferedLoadingController">
      <property name="buffer" value="2"/>
    </bean>
  </property>
  <property name="maxCacheSize" value="100MB"/>
  <property name="stale_interval" value="1000"/>
  <property name="timeout" value="-1"/>
  <property name="disabled" value="false"/>
  <property name="readonly" value="false"/>
</bean>
```

En la línea 2 se define el nombre de la capa teselada. Le sigue una descripción textual de la capa que aparecerá en los metadatos del servicio al invocar una petición *GetCapabilities*. El parámetro *url* contiene la dirección del servicio WMS remoto. En las líneas 5-7 se especifican los parámetros *layers*, *srs* y *format* que llevará la petición al servidor WMS. A continuación se define el perfil de teselado y los límites geográficos de la capa. En las líneas 10-23 se establece un gestor de caché con índice espacial de recopilación de estadísticas y almacén de teselas con persistencia en disco. En la línea 24 se activa el modo *debug* que superpone información sobre las imágenes de mapa, como las coordenadas de cada tesela y el número de peticiones recibidas. En las líneas 25-29 se habilita el *metatiling* con un tamaño de *buffer* de dos unidades (*metatile* 5×5). En la línea 30 se establece el tamaño máximo de la caché. En caso de alcanzarse este límite entre en juego una política de reemplazo. Finalmente, se definen los tiempos de expiración de las teselas y algunas propiedades para habilitar/deshabilitar la capa o establecerla en modo de sólo lectura.

Listado A.4: Configuración de las capas virtuales de tipo StatsLayerSet y CachedLayerSet.

```
<bean id="cartociudad_cached_layer" class="org.uva.idelab.wmscwrapper.layer.
    CachedLayerSet">
  <property name="targetLayer" ref="cartociudad_layer"/>
  <property name="name" value="cartociudad_cached"/>
  <property name="forward" value="false"/>
</bean>

<bean id="cartociudad_stats_layer" class="org.uva.idelab.wmscwrapper.layer.
    StatLayerSet">
  <property name="targetLayer" ref="cartociudad_layer"/>
  <property name="name" value="cartociudad_stats"/>
  <property name="useBackground" value="true"/>
</bean>
```

En el Listado A.4 se define un *Tileset virtual* que se genera al vuelo a partir de las estadísticas recopiladas por la caché de la capa. Aunque en el fondo es una herramienta para la depuración del funcionamiento de los algoritmos, demuestra una de las aplicaciones de generar teselas en tiempo real a partir de la información ya existente en la caché. Asimismo, se define una capa de tipo *CachedLayerSet* que muestra qué contenido está cacheado y cuál no.

A.2.3. Estrategia de monitorización

Las técnicas de gestión de las caché aplican heurísticas definidas para un dominio de aplicación. De esta manera se intenta maximizar la probabilidad de acierto $P_h\{T(i,j,n)\}$ a la vez que mantener el consumo de recursos dentro de los rangos definidos del sistema. Este equilibrio se logra mediante una adecuada política de limpieza de caché en la que los objetos con menor probabilidad de acceso $P_{req}\{T(i,j,n)\}$ son descartados [8] cuando es necesario liberar recursos.

Una estrategia popular es la heurística LRU en la que se supone que el elemento que hace más tiempo que no se ha solicitado es menos probable que sea accedido en lo sucesivo. Esta estrategia ha generado una familia de métodos que han funcionado bastante bien con páginas de memoria en microprocesadores o en documentos *Web* (Aggarwal et al., 1999). No obstante algunos investigadores como (Abrams et al., 1996) han demostrado que cuando los documentos tienen características heterogéneas (p.ej. su tamaño) las estrategias basadas únicamente en el patrón temporal de accesos resultan muy ineficaces.

Extrapolando estos resultados a nuestros objetos nos hace pensar que las heurísticas LRU pueden ser mejoradas teniendo en cuenta las características espaciales de las peticiones y las relaciones de adyacencia multidimensional de los objetos $T(i,j,n)$.

La actividad de la caché se monitoriza permanentemente almacenando los resultados en un índice espacial. Los mecanismos de gestión de la caché utilizan la información recogida en este índice para optimizar su funcionamiento.

A.2.4. Mecanismos de precarga de teselas y de limpieza de la caché

El módulo *wmscwrapper-jobs* incorpora mecanismos de mantenimiento de la caché que se ejecutan en segundo plano (*demonio*) incluso en ausencia de peticiones de usuarios. Las

[8] Realmente una heurística más inteligente descartaría los objetos de menor valor. El valor de un objeto es una composición de la probabilidad de ser demandado y otras características como puede ser el coste de producirlo (Abrams et al., 1996).

Figura A.9: Selección de zonas geográficas en ArcGis para la ejecución de una tarea de *seeding*.

otras implementaciones conocidas realizan las operaciones de mantenimiento en el mismo instante en que se realiza la petición de la tesela. La aproximación propuesta aquí pretende aumentar la QoS general de la caché durante las horas valle del servicio (ver Figura 5.10, pág. 57).

A.2.4.1. Selección manual de las regiones de interés

Como se comentó al comienzo de este trabajo, pregenerar todos los objetos de la caché implica importantes costes en cuanto a consumo de recursos de almacenamiento y tiempo de puesta en marcha del servicio.

Por ello, algunos de los sistemas de caché estudiados, como *TileCache* o *GeoWebCache*, ofrecen la posibilidad de indicar el *bbox* del conjunto de teselas que se quiere cachear durante el proceso de *seeding* para las escalas indicadas, para delimitar la zona de interés y precargar menos teselas. TileCache ofrece además la posibilidad de cachear las teselas inscritas en la circunferencia cuyo centro y radio se especifican.

Es inmediato anticipar que las aproximaciones anteriores ofrecen pocos grados de libertad al administrador, resultando ineficientes para el tratamiento de zonas geográficas más complejas. Para destacar las limitaciones existentes considérese por ejemplo la solución propuesta en ArcGis[9] para el cacheo de zonas geográficas de gran tamaño, como la mostrada en la Figura A.9, en la que se quieren evitar las zonas de agua. Obviamente se trata de una solución poco sofisticada, y no resulta óptima para el objetivo perseguido de incluir tan solo las zonas terrestres.

Como respuesta a las carencias detectadas, el sistema de caché WMSCWrapper incorpora la posibilidad de seleccionar geometrías arbitrarias. Esta aproximación permite acotar con mayor precisión las áreas de interés a cubrir por las tareas de mantenimiento de la caché, reduciendo el tiempo y los recursos necesarios para su ejecución. En la Figura A.10 se muestra la interfaz de usuario para la selección de estas zonas de interés.

Se han implementado en la caché una serie de iteradores para recorrer geometrías arbitrarias (puntos, líneas y polígonos), haciendo uso para ello de adaptaciones de los algoritmos tradicionales más comúnmente utilizados en el campo del renderizado gráfico, como son los algoritmos de Bresenham (Bresenham, 1965) y ScanLine (Lane et al., 1980) para el recorrido de líneas y polígonos, respectivamente.

[9] http://webhelp.esri.com/arcgisserver/9.2/dotNet/manager/publishing/tips_map_caches.htm

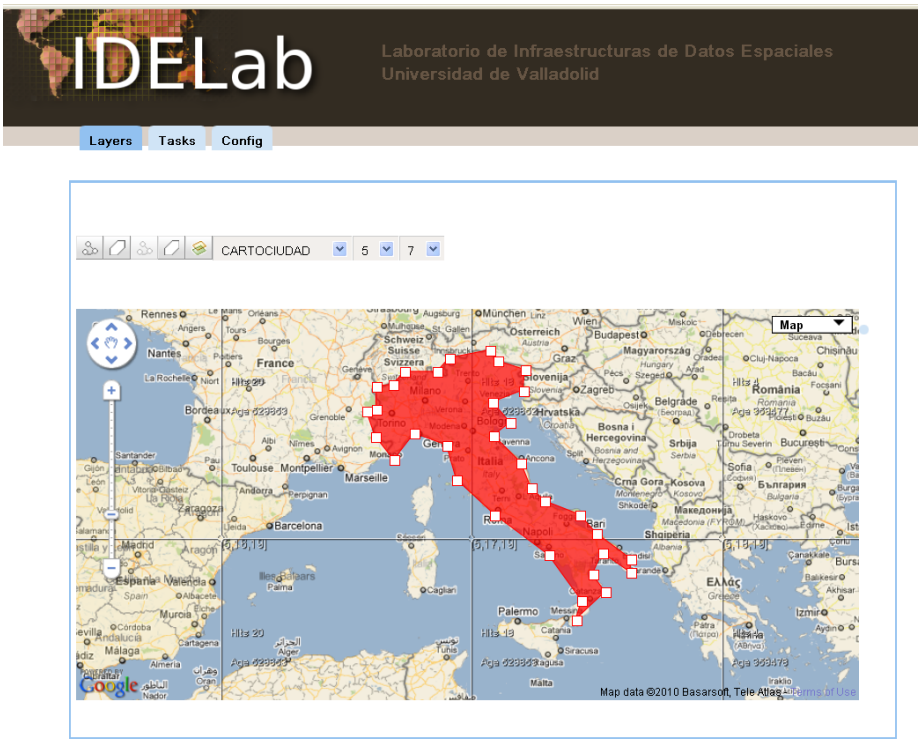


Figura A.10: Interfaz gráfica de usuario para la selección de geometrías en el mapa.

Los mecanismos de mantenimiento de caché utilizan estos iteradores para ejecutar la labor concreta sobre la zona geográfica especificada, ya sea para el cacheo de teselas o el truncado de las mismas en las respectivas tareas de *seeding* o limpieza, o para la actuación de cualquier otro mecanismo de gestión.

A.2.4.2. *Seeder predictivo basado en* features

Sin embargo, las soluciones anteriores son completamente manuales y se basan en la intuición del administrador. Sería más interesante el disponer de mecanismos automáticos o semiautomáticos para la identificación avanzada de regiones potencialmente candidatas a ser generadas durante el proceso de *seeding*, o borradas de la caché por las tareas de limpieza o reemplazo.

WMSCwrapper incorpora una sencilla implementación para la experimentación con estrategias semi-automáticas de generación predictiva. Sobre este banco de trabajo se ha desarrollado un *plug-in* que utiliza distintas fuentes externas de información para la generación de *heatmaps* o zonas calientes, como se muestra en la Figura A.11. Estos *heatmaps* pueden interpretarse como mapas de estimación de la probabilidad de acceso a las teselas de mapas, en los que las zonas calientes tienen asociada una mayor probabilidad de ser pedidas. Los mecanismos de mantenimiento de la caché utilizan estos *heatmaps* para la mejora de su funcionamiento.

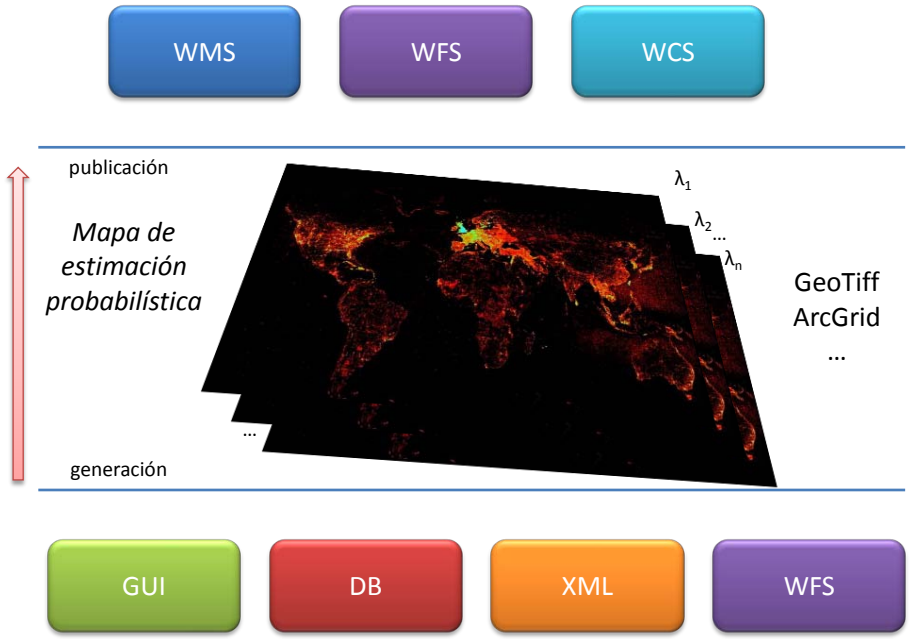


Figura A.11: Generación y consumo de *heatmaps* para explotación en la caché.

A.2.4.3. *Web Feature Service* - Aproximación *Up-Down*

Como se introdujo en el Capítulo 3, dada la naturaleza exponencial de la estructura de datos manejada en una caché con estructura en pirámide de escalas, en la que el número de elementos aumenta exponencialmente con el nivel de resolución, la aplicación de algoritmos analíticos y predictivos se puede tornar impráctico, incluso mediante el apoyo de mecanismos heurísticos. Por ello la aproximación más viable es la de trabajar con medidas recopiladas en un cierto nivel intermedio de la pirámide y posteriormente extrapolar esta información a los niveles inferiores. A partir de ahora se utilizará el término *Up-Down* para referirse a esta aproximación. En el desarrollo experimental realizado, la información recopilada en este nivel intermedio de la caché se almacena en un *heatmap* que puede ser posteriormente utilizado por diversos mecanismos de mantenimiento en la caché.

En la Figura A.12 se ilustra el procedimiento seguido para la generación de un *heatmap* según la aproximación *Up-Down*, a partir de una determinada *feature* (podría representar p.ej. un río, o un vial) accesible a través de un servicio OGC estándar *Web Feature Service* (WFS). Se recorren las teselas del nivel intermedio realizando una consulta espacial acerca de la presencia o no de dicha *feature* (ó de cualquier propiedad de esta: área, longitud, importancia, etc) en el *Bounding Box* que abarca cada tesela. En la especificación WFS 1.1 (Vretanos, 2005) se ha añadido el parámetro opcional *resultType=hits* para permitir al servidor devolver un contador de cuántas *features* coinciden con la petición realizada. De esta forma se ofrece una opción más eficiente para la obtención del número de *hits* o aciertos, sin tener que solicitar un documento completo con los fenómenos que cumplen la consulta y posteriormente contabilizarlos.

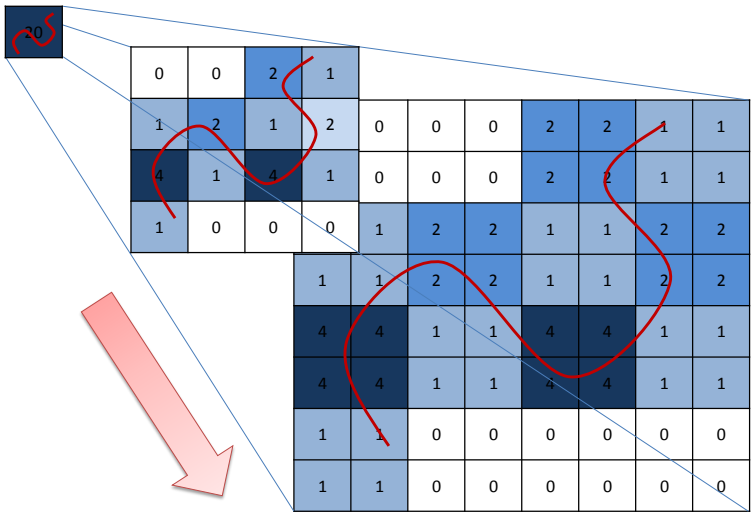


Figura A.12: Estrategia de generación de *heatmap* a través de las *features* de un WFS según la aproximación *up-down*.

A.2.4.4. *Web Feature Service* - Aproximación *Bottom-Up*

En el caso de que la información a considerar sea dispersa la aproximación *Up-Down* no resulta muy eficiente. Caractericemos formalmente este concepto de «dispersión» de una *feature*. Una determinada *feature* se considera dispersa en un determinado nivel de resolución si el número de celdas por las que se extiende es mucho menor que el número total de celdas de dicho nivel. En estos casos, en vez de recorrer todas las teselas para ver si abarcan la *feature* de interés, resulta más apropiado recorrer estas últimas y determinar qué teselas la contienen, propagando esta información hacia niveles superiores. A partir de ahora nos referiremos a esta aproximación como *Bottom-Up*.

En la Figura A.13 se muestra el procedimiento para la generación de un *heatmap* según la aproximación *Bottom-Up*.

La aplicación de esta aproximación en escenarios de producción, en los que una capa puede tener miles de *features*, puede dar lugar a múltiples fuentes de error. Generalmente, los servidores WFS tienen un límite en el número máximo de *features* a devolver en una sola *query*, truncando el resultado de una operación *getFeature* a este valor y dando lugar a respuestas WFS incompletas, sin devolver todas las *features* solicitadas. En Geoserver, al igual que en otros servidores con soporte WFS, este límite es configurable por el usuario[10]. Sin embargo, incrementar este valor de forma indiscriminada con el objetivo de alcanzar el límite deseado desencadena *Timeouts* en el lado del cliente, fruto de las elevadas latencias para obtener un número muy elevado de *features*, o desbordamiento de *buffers* de memoria, agravando el problema.

La solución más apropiada sería disponer de un mecanismo que permitiese realizar *paginado* de las peticiones. Geoserver introduce el concepto de paginado para el servicio WMS[11], pero no ofrece esta funcionalidad en el servicio WFS, a pesar de que su aplicación es

[10] http://geoserver.org/display/GEOSDOC/Server+Configuration
[11] http://geoserver.org/display/GEOSDOC/WMS+vendor+parameters

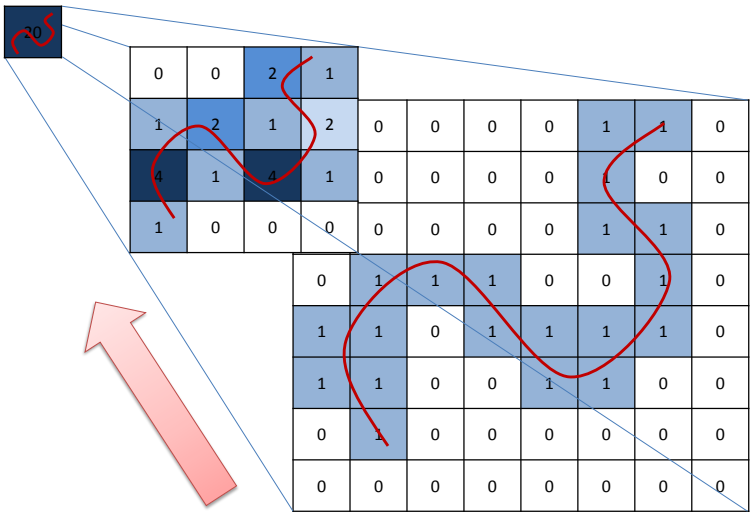


Figura A.13: Estrategia de generación de *heatmap* a través de las *features* de un WFS según la aproximación *bottom-up*.

más propia de este último. En el caso de WMS esta funcionalidad adquiere mayor relevancia en el caso de formatos vectoriales, como KML y SVG, permitiendo devolver un mapa con un elevado número de *features* de forma progresiva. El paginado WMS en Geoserver se consigue mediante la incorporación de dos parámetros en la petición de mapas:

- STARTINDEX: Un entero positivo que especifica el índice de comienzo dentro de una lista ordenada de *features*.
- MAXFEATURES: Valor entero que establece un límite en la cantidad de *features* a presentar.

Esta funcionalidad de paginado no está recogida en la especificación WFS de OGC (Vretanos, 2005), y es una carencia que ya se ha detectado y recogido en (Rushforth, 2007).

Dada esta carencia, se ha propuesto una solución alternativa que, aunque menos óptima, satisface las necesidades indicadas. El procedimiento seguido consiste en dividir sucesivamente la región a cubrir hasta que cada región individual contenga un número de *features* menor que un límite dado (configurable). Para averiguar de forma eficiente el número de *features* albergadas en cada región se utiliza el parámetro opcional *resultType=hits* de la petición WFS comentado anteriormente.

Mediante esta aproximación pueden obtenerse capas vectoriales con un número arbitrario de *features*. Para la validación de esta estrategia se ha ejecutado una tarea de *seeding* utilizando la *feature* que contiene los segmentos de los ríos de la IDE de la Confederación Hidrográfica del Duero (CHDuero)[12]. Esta capa vectorial está compuesta por 2016 *features* de tipo *LineString* con un elevado número de coordenadas (del orden del millar). En la Figura A.14 se muestra una representación a gran escala de esta capa vectorial. Por otra parte, en la Figura A.15 se muestra una captura de la capa *CachedLayerSet* para visualizar el contenido cacheado tras precargar dicha capa según la aproximación *bottom-up* mediante el algoritmo de Bresenham.

[12]http://www.chduero.es/

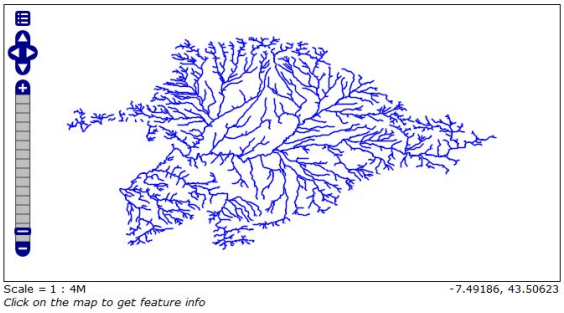


Figura A.14: Capa vectorial con los segmentos de los ríos de la IDE de la Confederación Hidrográfica del Duero.

A.2.4.5. *HeatMap creado a partir de los registros de peticiones a servidores de mapas*

Partiendo de la premisa de que puede realizarse una estimación de la probabilidad de acceso futuro a las teselas atendiendo a la información disponible en accesos pasados, los registros de acceso o *logs* de los servidores de mapas suponen una valiosa fuente de información. De entre la información disponible en los registros de peticiones no se ha considerado la identidad del cliente, puesto que esta sería necesaria tan solo para la realización de una predicción de accesos particularizada por usuario.

Se han implementado sendos generadores para la obtención de mapas de estimación probabilística de las peticiones de los usuarios, mediante las aproximaciones *Up-Down* y *Bottom-Up* ya comentadas, tomando los registros almacenados en base de datos como fuente de información.

En la Figura A.16 se muestra el mapa de estimación de probabilidades *a posteriori* de las peticiones de los usuarios al servicio WMS-C de Cartociudad. Se observa una mayor concentración de peticiones realizadas sobre territorios como Madrid, Valladolid, Zaragoza o Barcelona. Estas zonas «calientes» o de gran probabilidad son buenas candidatas a ser cacheadas por una tarea de *seeding*, y los algoritmos de limpieza y actualización deberían mantener la cartografía de estas regiones en caché.

A.2.4.6. Almacenamiento de los *heatmaps*

Se han definido nuevos mecanismos interoperables para la recopilación de estadísticas, utilizables como fuentes de datos para los algoritmos de análisis y gestión.

A partir de los índices espaciales presentes en WMSCWrapper, se han implementado nuevos almacenes de tipo *Grid Coverage* para el almacenamiento de los *heatmaps* generados, así como la posibilidad de obtener un almacén de este tipo a partir de los ya existentes. Un *Grid Coverage* es una estructura tipo *raster* que contiene una matriz de valores numéricos junto con información acerca del significado de estos valores. Además, este *raster* está geo-referenciado. De momento, se permite el almacenamiento en los populares formatos GeoTIFF[13] y ArcGrid. Muchos servidores de datos geoespaciales ofrecen la posibilidad de utilizar estos formatos como orígenes de datos *raster* para su publicación a través de servicios OGC estándar. Esto posibilita, por ejemplo, el uso del servicio WMS para obtener

[13] http://trac.osgeo.org/geotiff/

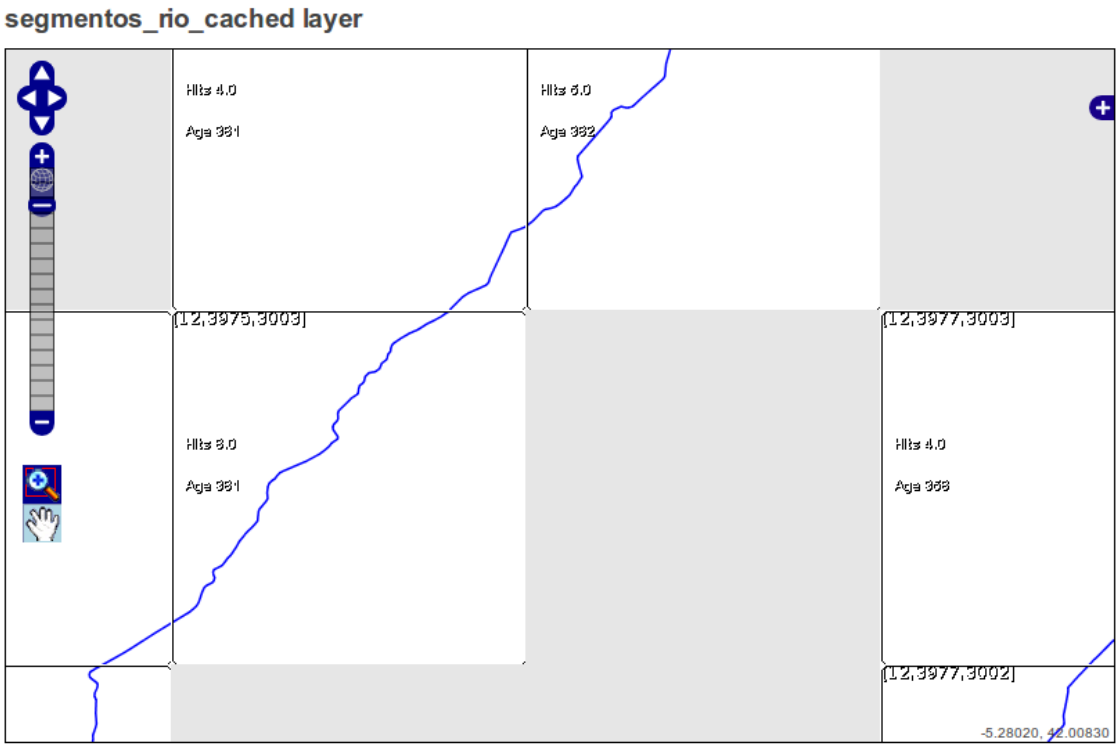


Figura A.15: Capa *CachedLayerSet* con el contenido cacheado tras ejecutar una tarea de precarga con la aproximación *Bottom-Up*. Las teselas sombreadas no están en caché.

una representación visual de los *heatmaps*, como la mostrada en la Figura A.16, con la posibilidad de personalizar la visualización mediante *Styled Layers Descriptor* (SLD)[14].

Por otra parte, los mecanismos de caché pueden beneficiarse de todas las posibilidades ofrecidas por el servicio *Web Coverage Service* para acceder a los valores numéricos almacenados en los *heatmaps*.

La solución adoptada resulta muy adecuada para satisfacer el objetivo de ofrecer la posibilidad de acceder a estos datos mediante servicios estándar, lo que garantiza el acceso interoperable a los mismos. Un servicio WMS puede, por tanto, publicar sus *heatmaps* a través de dichas interfaces, de forma que los *proxy cache* que utilizan estos servicios puedan beneficiarse de esta información.

A.2.4.7. Procesamiento de los *heatmaps*

Por otra parte, al recopilar la información en formato de imagen pueden aprovecharse las herramientas de procesado de imágenes disponibles. Pueden utilizarse, por ejemplo, herramientas para la detección de cambios abruptos en la frecuencia de la imagen (cambios en el gradiente de la intensidad de la misma), o herramientas de *clustering* para el particionado de elementos en distintos grupos en base a su semejanza. De esta forma, mediante procesado de imágenes pueden extraerse regiones de interés (ROI - *Region Of Interest*), candidatas para la realización de tareas de mantenimiento de la caché.

[14] http://www.opengeospatial.org/standards/sld

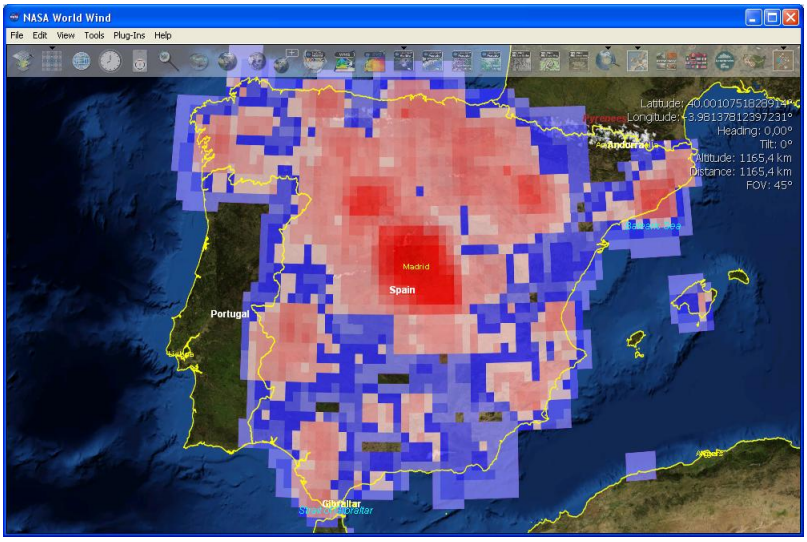


Figura A.16: Mapa de estimación de probabilidad *a posteriori* calculado a partir de los registros de Cartociudad. Accedido por un cliente de mapas *WorldWind* a través de un servicio WMS estándar.

Considérese el *heatmap* mostrado en la Figura A.16. Este mapa muestra la probabilidad estimada de acceso a las zonas geográficas representadas por las teselas. Las zonas en rojo corresponden a teselas con gran probabilidad de ser pedidas, mientras que las zonas en azul corresponden a teselas cuya probabilidad estimada de ser pedidas es menor. Esta información puede ser utilizada por un algoritmo de *seeding* como el descrito en el Capítulo 7.2, cacheando primero las teselas con mayor probabilidad de ser pedidas. Podría utilizarse de forma semejante para otros mecanismos, como los de limpieza o actualización de la caché. Para ello es necesario identificar en primer lugar las regiones homogéneas de probabilidad.

Una posible forma de hacerlo es mediante la detección de cambios abruptos en el gradiente de la intensidad de la imagen para la identificación de ROI. En la Figura A.17 se muestra una imagen procesada en la que se ha utilizado la identificación de cambios abruptos en la frecuencia de la imagen para la detección de flancos. Estos flancos pueden utilizarse posteriormente como máscaras para la selección de las regiones consideradas de interés.

También se ha experimentado en este banco de pruebas con algoritmos de *clustering* para la definición de regiones de interés. *Clustering* es el proceso de agrupar un conjunto de datos de forma que se maximice la similitud entre aquellos del mismo *cluster* y se minimice la similitud entre los que pertenecen a *clusters* distintos (Wang et al., 2006). A continuación se presenta una definición formal del problema:

Dado un conjunto de datos $X = \{x_1, ..., x_N\}$, donde N es el número de datos. Sea u_{ij} la función de pertenencia del elemento i-ésimo del vector $(i = 1, ..., N)$ al j-ésimo *cluster* $(i = 1, ..., N)$, donde C es el número de *clusters*. Generalmente se aplican las siguientes restricciones a los valores de pertenencia:

$$\forall i, \sum_{j=1}^{C} u_{ij} = 1; \forall i, j, u_{ij} \in [0, 1]; \forall j, \sum_{i=1}^{N} u_{ij} > 0. \tag{A.3}$$

El criterio de *clustering* más ampliamente utilizado es el de minimización de la función

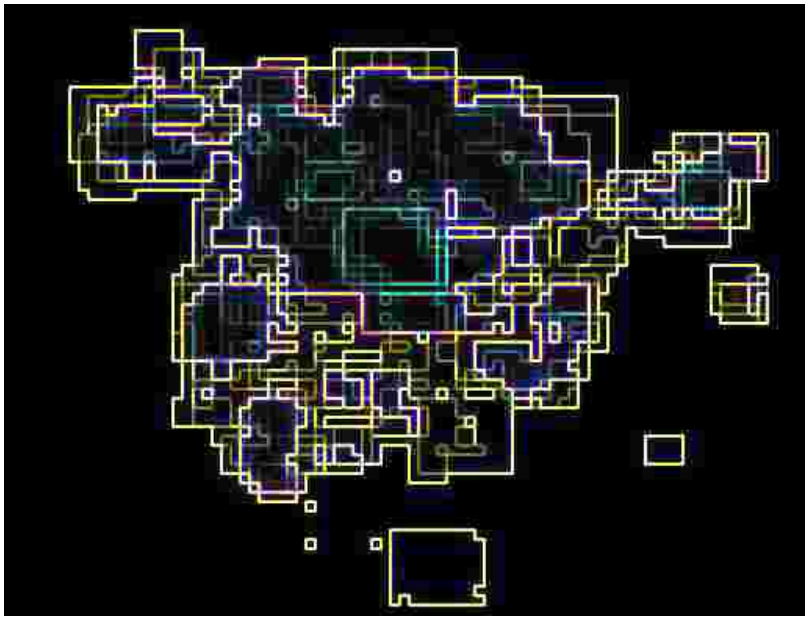


Figura A.17: Identificación de cambios de gradiente mediante procesado de imagen para la generación de ROI (*Regions of Interest*).

objetivo

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{i,j}^{m} \left\| x_i - v_j \right\|^2 . \tag{A.4}$$

Donde $V = v_1, v_2, ..., v_C$ es el vector con los centros de los *clusters* y m es un coeficiente que determina el grado en el que miembros pertenecientes a un determinado *cluster* afectan el valor de los centroides.

Este algoritmo, aplicado sobre el mapa de probabilidad, divide los N *píxels* de la imagen en C *clusters*, de forma que en cada *cluster* se agrupan los *píxels* de intensidad semejante. En la Figura A.18 se muestra la división en 4 *clusters* del *heatmap* mostrado en la Figura 5.11b (página 58). Podemos interpretar cada *cluster* como un conjunto de teselas cuya probabilidad de ser pedidos es muy similar. Esta agrupación de las teselas en *clusters* facilita en gran medida el funcionamiento de los algoritmos de gestión.

A.2.5. Benchmarking del servicio de caché

Para valorar objetivamente las mejoras de rendimiento conseguidas mediante la adopción de un sistema de caché se ha recurrido al uso de *benchmarks*. Estos *benchmarks* permitirán cuantificar las mejoras de rendimiento alcanzadas fruto de la optimización de los algoritmos desarrollados sobre la caché WMSCWrapper. Durante el análisis del estado del arte no se ha detectado la existencia de ningún estudio comparativo exhaustivo de tales sistemas de caché. En (Ángel Esbrí Palomares y Valero, 2005) y en (P. et al., 2008) se presentan resultados de *benchmarks* para la medida de rendimiento de servidores de mapas, pero no para sistemas de caché. Asimismo, OSGeo hace un reporte anual en las conferencias FOSS4G de los resultados de *benchmarks* de los principales servidores de mapas WMS

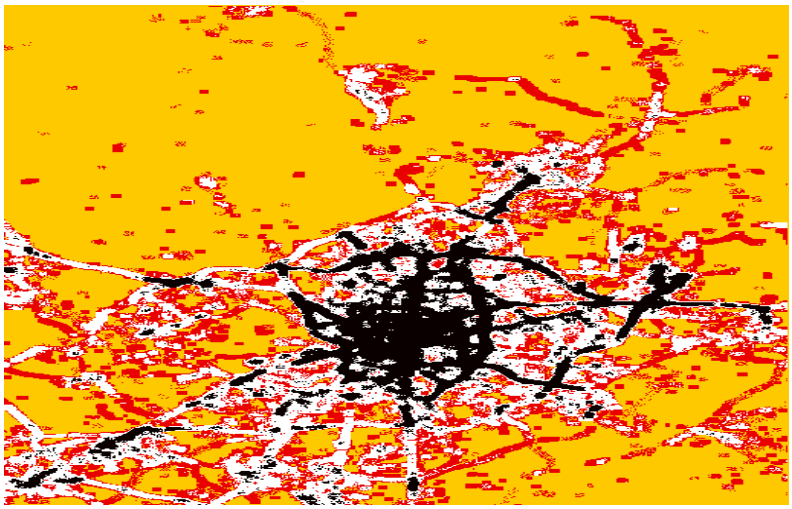


Figura A.18: Detección de Regiones de Interés (RoI) mediante *clustering*. Agrupación en 4 *clusters*

(Aime y McKenna, 2009). En todos estos estudios se utilizan peticiones «sintéticas», generadas aleatoriamente en la mayoría de los casos, o incluso se repite continuamente una misma petición. Obviamente, estos patrones de peticiones no se corresponden con el perfil de peticiones de los usuarios en un sistema real en un ámbito de *producción*, con lo que los resultados obtenidos pueden no ser del todo representativos de lo que sucede en un escenario más realista.

A continuación se describe el banco de pruebas desarrollado para las medidas de rendimiento. Asimismo, se presentan los resultados de un estudio comparativo del rendimiento de distintos sistemas de caché WMS: GeoWebCache[15], TileCache[16] y WMSCWrapper[17].

Descripción del banco de pruebas

Con el objetivo de plantear un escenario más cercano a la realidad para la realización de medidas de rendimiento sobre distintos sistemas de caché de teselas, se plantea aquí el uso de los registros disponibles con las peticiones realizadas sobre el servicio WMS-C de Cartociudad. De esta manera, el patrón de peticiones utilizado representa fielmente el de un escenario real.

El entorno de pruebas para la realización de las medidas de rendimiento de las diversas cachés es el mostrado en la Figura A.19. Los componentes localizados en el interior de la línea punteada están alojados en un servidor local, mientras que el resto de componentes están alojados en servidores remotos. Como servidor WMS se ha utilizado el servidor web de mapas de Cartociudad.

La herramienta de pruebas JMeter[18] está alojada en una misma máquina junto a las cachés de mapas, y se utiliza la interfaz de red *localhost*. De esta forma se independizan los resultados de rendimiento de las cachés de la red utilizada, evitando que una posible

[15] http://geowebcache.org
[16] http://tilecache.org
[17] http://dev.idelab.uva.es/wmscwrapper
[18] http://jakarta.apache.org/jmeter/

saturación de la conexión de red invalide los resultados obtenidos. El servidor local es un Intel Pentium(R) Dual-Core E5300 a 2.60GHz con 4GB de memoria RAM, con Sistema Operativo Ubuntu 9.10, Kernel de Linux 2.6.31-14-generic.

Se ha utilizado TileCache v.2.10 corriendo sobre un servidor Web Apache 2.0 bajo mod_pyhton (se ha comprobado que esta instalación de la caché es la que obtiene un mejor rendimiento). En cuanto a las implementaciones Java, se ha utilizado GeoWebCache v.1.2.1 y WMSCWrapper v.1, desplegadas sobre un servidor de aplicaciones Tomcat 6 con máquina virtual de Java de 64 bit (JDK 1.6.0_15).

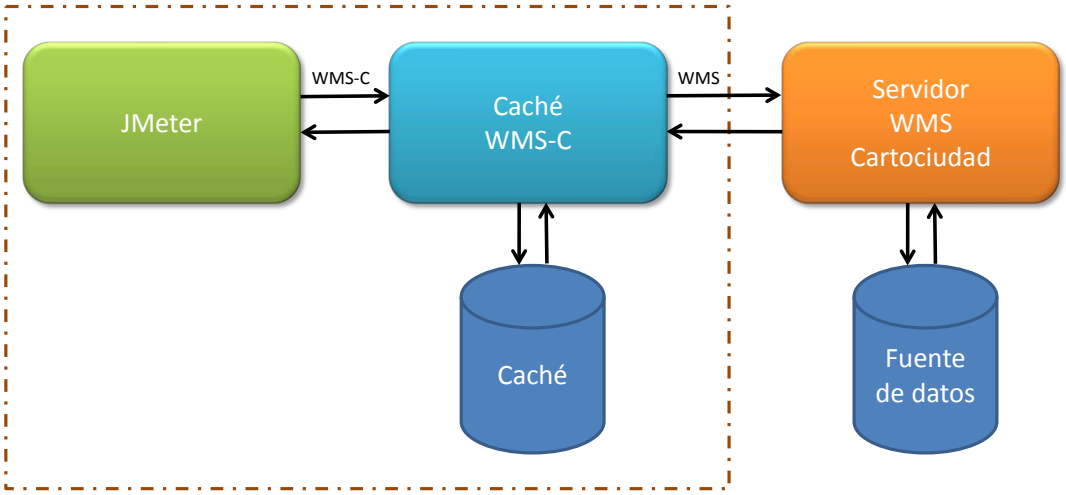


Figura A.19: Arquitectura del entorno de pruebas

Generación de los bancos de pruebas

Para la generación de los *bboxes* de las peticiones WMS se ha desarrollado una herramienta en Java que ofrece múltiples posibilidades:

- Utiliza una base de datos como fuente de peticiones. Pueden almacenarse en base de datos los registros de cualquier servicio de mapas y posteriormente utilizar estos registros para la reproducción de un escenario real.
- Se puede restringir el área geográfica de las peticiones, así como especificar las escalas de resolución deseadas.
- Posibilidad de generación «ordenada» (en el dominio del tiempo) de las peticiones, o desordenarlas de forma aleatoria.

Tras la ejecución de la herramienta desarrollada se obtiene un fichero .csv que contiene los *bboxes* de las peticiones, que puede ser directamente interpretado por JMeter como fuente de peticiones. En la Figura A.20 se muestra una captura del plan de pruebas de JMeter, donde se ilustra la definición de los parámetros de las peticiones WMS-C.

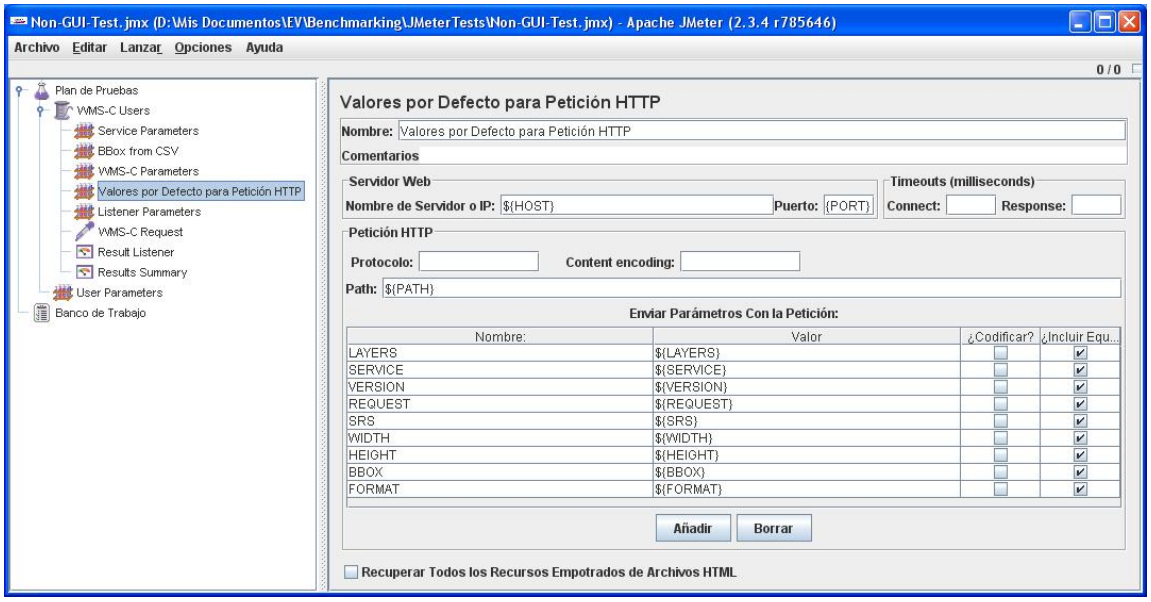


Figura A.20: Definición de los parámetros de la petición WMS-C en JMeter

Resultados

En la Figura A.21 se recogen los resultados obtenidos de las medidas de rendimiento en términos de tasa de transmisión o *throughput* (teselas/segundo) para un escenario en el que todas las peticiones están en caché (mejor caso), en función del número de usuarios simultáneos. A partir de los resultados obtenidos se extrae que la caché WMSCWrapper ofrece un mayor rendimiento que sus competidoras en el escenario bajo test.

En la Figura A.22 se recogen los resultados correspondientes al escenario en el que la caché está inicialmente vacía, produciéndose por tanto fallos en caché. Del total de 312.382 peticiones realizadas, un 62.23 % de las peticiones suponen aciertos (*hits*) en caché, correspondiendo el 37.78 % restante de peticiones a fallos (*misses*) en caché. Cada fallo de caché implica una petición al servidor WMS remoto, con el tiempo de generación y transmisión de la imagen incurridos por parte de este. Por este motivo, se aprecia una disminución muy significativa en el rendimiento conseguido por las diversas implementaciones.

Como se comentó en el Capítulo 8, frecuentemente se utiliza la técnica de generación de peticiones de mayor tamaño que la tesela a cachear o *metatiling*, y posteriormente se postprocesa para aprovechar la información disponible y generar nuevas teselas. De esta forma se reduce el número de consultas al WMS remoto.

La aplicación de esta estrategia durante una tarea de *seeding* puede incrementar significativamente el rendimiento conseguido. Para la realización de las medidas de rendimiento de esta técnica se ha utilizado un conjunto de *bboxes* ordenados espacialmente que cubren la península ibérica en su totalidad desde la resolución 10 a la 15, utilizando como capa la conjugación de las capas *DivisionTerritorial*, *FondoUrbano*, *Vial*, *Portal*, *Toponimo*, *SeccionCensal* y *CodigoPostal* del servidor WMS de Cartociudad. En las Figuras A.23 y A.24 se muestran los tiempos de respuesta y tasas de transmisión medios para distintos tamaños de *metatile*, respectivamente. Se observa que, a pesar de que aumenta la latencia por la creación y procesamiento de una imagen de mayor tamaño, la latencia media de las

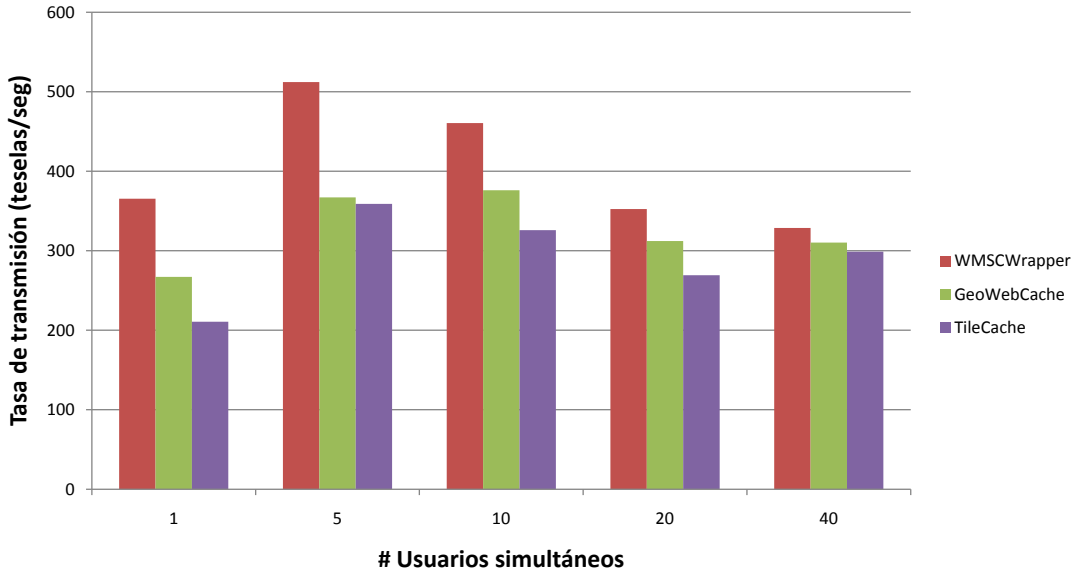


Figura A.21: Resultados de las medidas de rendimiento en términos de *throughput* de los sistemas de caché TileCache, GeoWebCache y WMSCWrapper para distinto número de hilos. Caché inicialmente llena.

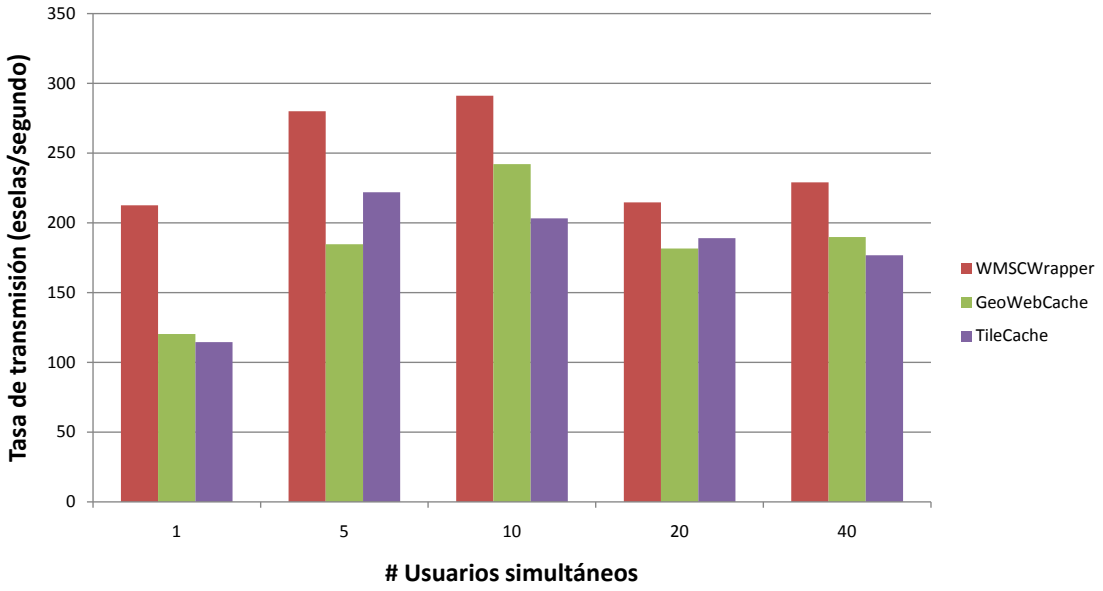


Figura A.22: Resultados de las medidas de rendimiento en términos de *throughput* de los sistemas de caché TileCache, GeoWebCache y WMSCWrapper para distinto número de hilos. Caché inicialmente vacía.

peticiones se reduce, pues disminuye el número de peticiones al servidor WMS remoto.

Un factor limitante a la hora de decidir el tamaño de *metatile* a utilizar es la cantidad de memoria que requiere la generación de la imagen.

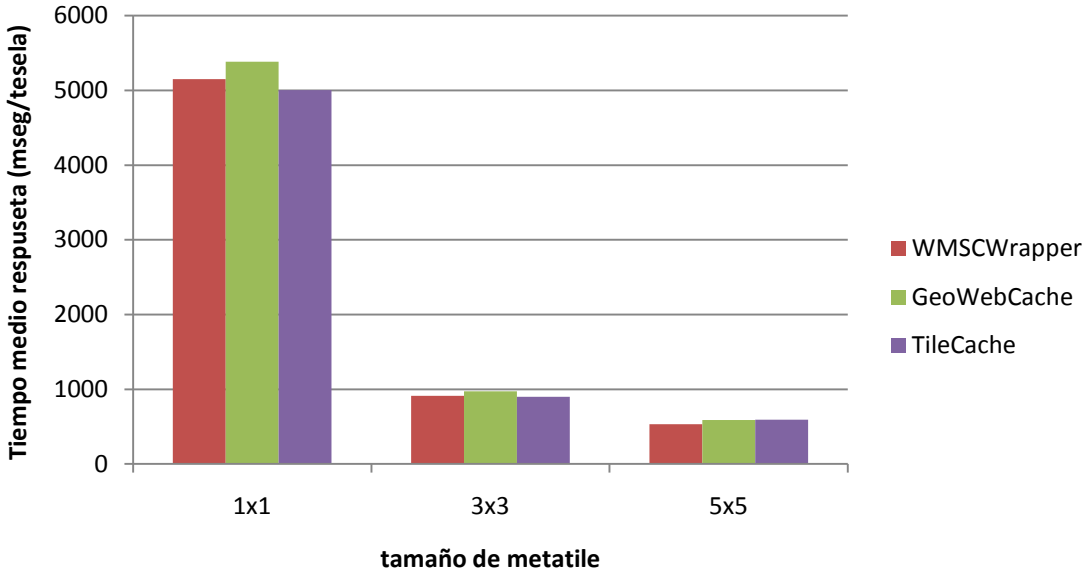


Figura A.23: Tiempos de respuesta para distintos tamaños de *metatile*, con caché inicialmente vacía.

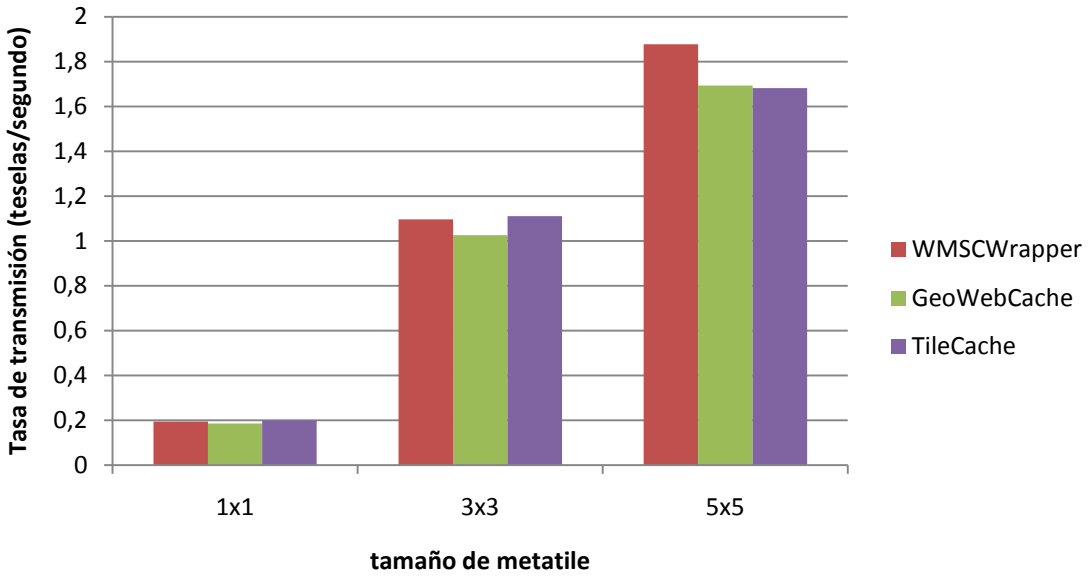


Figura A.24: Tasa de transmisión para distintos tamaños de *metatile*, con caché inicialmente vacía.

A.3. Extensión del prototipo de caché de teselas GeoWebCache

Los contenidos del presente apartado han sido parcialmente publicados en (García et al., 2012b).

GeoWebCache es una caché de teselas de código abierto, desarrollada por OpenGEO bajo licencia LGPL (*GNU Lesser General Public License*). Esta caché está integrada en GeoServer, aunque también está disponible como un producto *standalone* para su uso con otros servidores de mapas.

Es compatible con la especificación WMS mediante la recombinación y re-muestreo de teselas para servir peticiones de mapa arbitrarias no-restringidas a ninguna rejilla. Además, implementa el servicio WMTS de OGC, así como las especificaciones WMS-C y TMS. Asimismo, ofrece también soporte para clientes propietarios como Google Maps, Microsoft Bing Maps, Yahoo Maps, y Google Earth mediante *overlays* de KML.

Ofrece un mecanismo de precarga muy básico que permite especificar la zona a cachear mediante la especificación de su *Bounding Box* y el rango de resoluciones. Esta tarea se puede configurar mediante la herramienta de administración Web de GeoWebCache, o bien a través de una interfaz REST. Dadas las limitaciones encontradas en la definición de una tarea de *seeding*, a continuación se describe una extensión de GeoWebCache para la selección de zonas de interés mediante el modelo predictivo propuesto en el Capítulo 7.3.

GeoWebCache dispone de un sistema de almacenamiento que consta de dos componentes; El primero de ellos, *blobstore*, es un mecanismo de almacenamiento para las teselas en disco. El segundo, conocido como *metastore*, es un componente opcional para el almacenamiento de información relativa a las teselas, como el tiempo de creación de cada tesela y su tamaño de almacenamiento. Este componente permite controlar la expiración de las teselas en caché en caso de que se haya especificado un tiempo de vida. Asimismo, permite vigilar que el tamaño de la caché no supere los límites de almacenamiento establecidos.

A.3.1. Implementación de la solución

GeoWebCache está desarrollado utilizando la herramienta de gestión de proyectos Maven, lo cual agiliza el proceso de desarrollo del proyecto. Además, al hacer uso del *framework* de desarrollo Spring, se facilita la incorporación de nuevos componentes al sistema.

El proyecto se compone de un proyecto padre y varios módulos hijos, como se muestra en la Figura A.25. El módulo `gwc-web` contiene los elementos propios para la generación de la aplicación Web. Sin embargo, por defecto está configurado para generar un empaquetado *jar*, y no *war*.

Para implementar la solución propuesta se han añadido 3 nuevos módulos a GeoWebCache (ver Figura A.26)[19]:

- `gwc-stats`: Almacenamiento de las estadísticas de acceso a las teselas.
- `gwc-featurecatalog`: Catálogo de fenómenos geográficos.

[19]Puede accederse al sitio Web del desarrollo de este desarrollo en `http://mvn.idelab.uva.es/gwc-modules`.

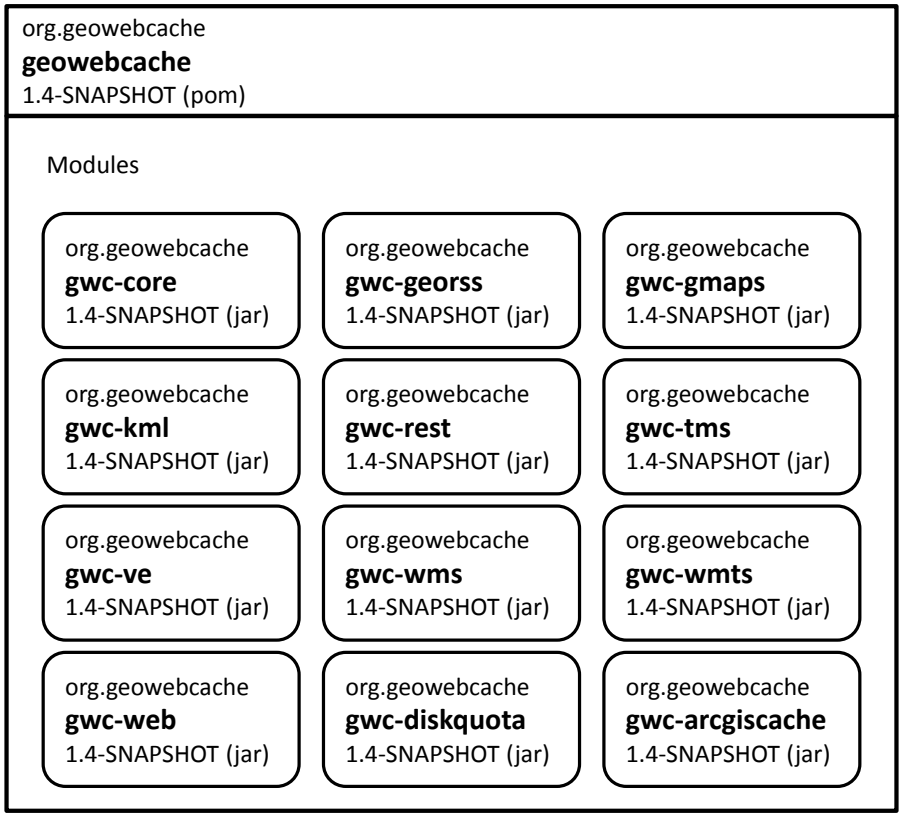


Figura A.25: Organización del proyecto Maven de GeoWebCache.

- `gwc-prefetch`: Calcula la solución OLS que mejor ajusta los fenómenos del catálogo con las estadísticas almacenadas.

El módulo `gwc-idelab` es una réplica del módulo `geowebcache` original, sustituyendo sus módulos hijos por sus respectivas dependencias de Maven, y añadiendo los 3 módulos nuevos. Asimismo, el módulo Web `gwc-web` ha sido sustituido por el módulo `gwc-idelab-web`. Las diferencias con el anterior es que genera un empaquetado *war*, y añade dependencias a los tres nuevos módulos.

A.3.2. Módulo para el almacenamiento de estadísticas

El módulo `gwc-stats` dota a GeoWebCache de la posibilidad de almacenar metadatos de las teselas: tiempo de creación, espacio de almacenamiento en disco, tiempo del último acceso, tiempo de modificación y número de accesos.

En la Figura A.27 se muestran los componentes de este módulo; En caso de estar habilitado, el componente `statsStore` asocia un escuchador (`tileStatsListener`) a las capas dadas de alta en GeoWebCache (ver Figura A.28). De esta forma, cada petición recibida por una capa desencadena una llamada al escuchador indicando qué tesela ha sido solicitada. A su vez, éste actualiza el índice en el que se almacenan las estadísticas (`statsIndex`), insertando la información relativa a esta tesela si es la primera vez que se pide, o actualizando su contador y tiempo de último acceso, en caso contrario. Para

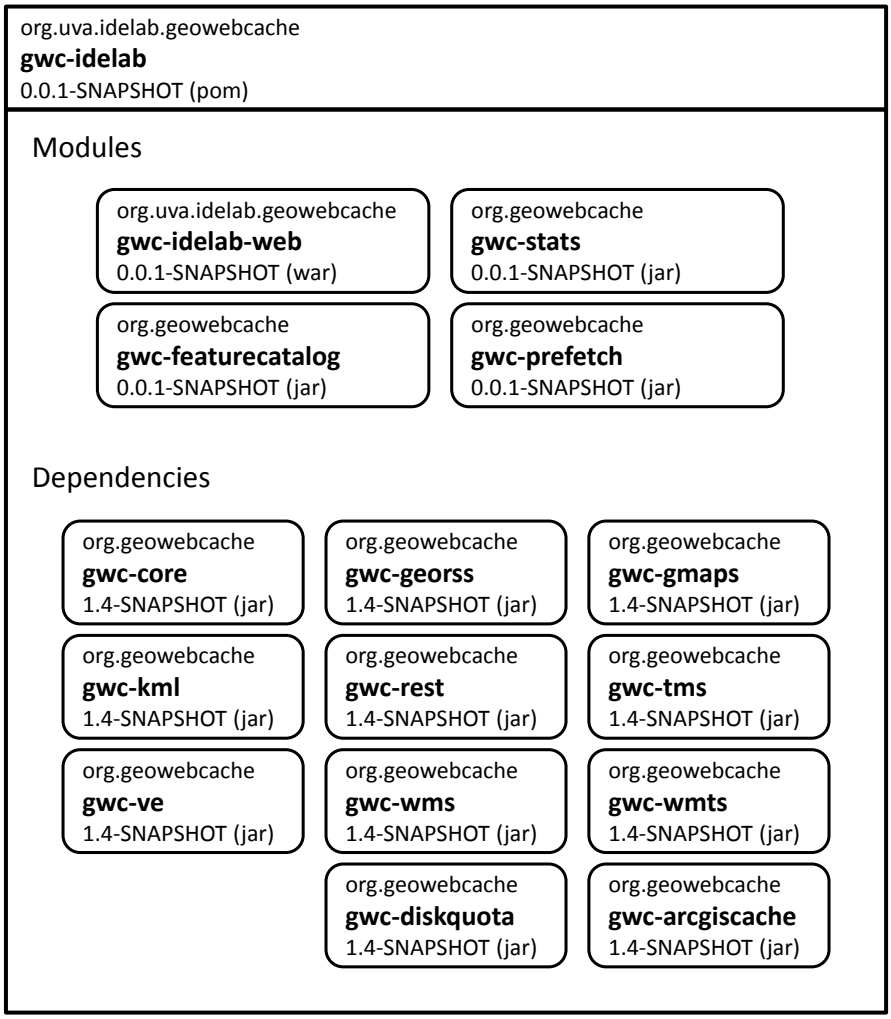


Figura A.26: Organización del proyecto Maven para la implementación propuesta.

el almacenamiento de estadísticas se utilizan los *dataSource* de GeoTools. Por defecto, el índice se almacena en una base de datos H2 en memoria con soporte para índices espaciales mediante HatBox. También se ha implementado la posibilidad de almacenar el índice en una base de datos PostgreSQL con la extensión espacial PostGIS. En este último caso es necesario especificar los parámetros de acceso a la base de datos.

En el índice de estadísticas puede configurarse el rango de niveles de resolución que se quiere registrar. Se puede configurar un factor de simplificación para agrupar las estadísticas de teselas vecinas y así reducir la sobrecarga derivada del almacenamiento de las mismas.

A.3.3. Módulo de catálogo de fenómenos geográficos

El módulo `gwc-featurecatalog` dota a GeoWebCache de un catálogo al que pueden añadirse colecciones arbitrarias de fenómenos geográficos. El uso de GeoTools facilita y ofrece gran versatilidad para la población del catálogo (ver Figura A.29). Las colecciones de fenómenos (*FeatureCollection* de GeoTools) pueden obtenerse realizando una consulta

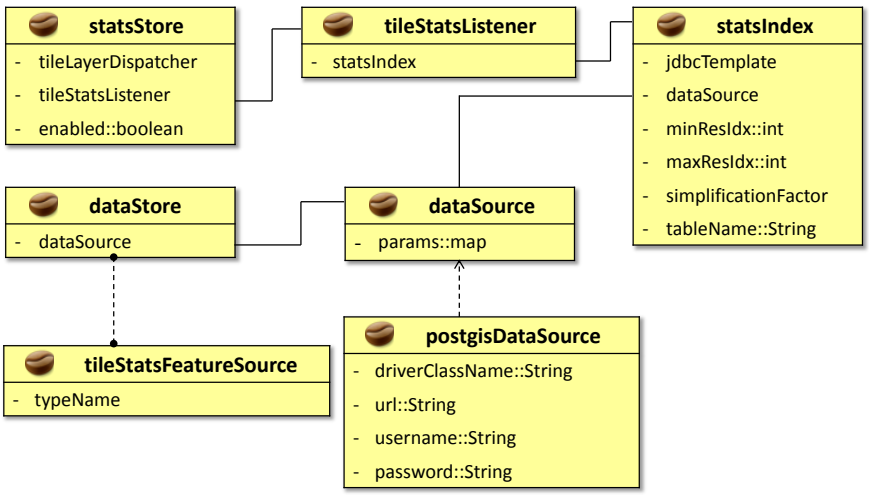


Figura A.27: Componentes principales del módulo gwc-stats (fichero geowebcache-stats-context.xml).

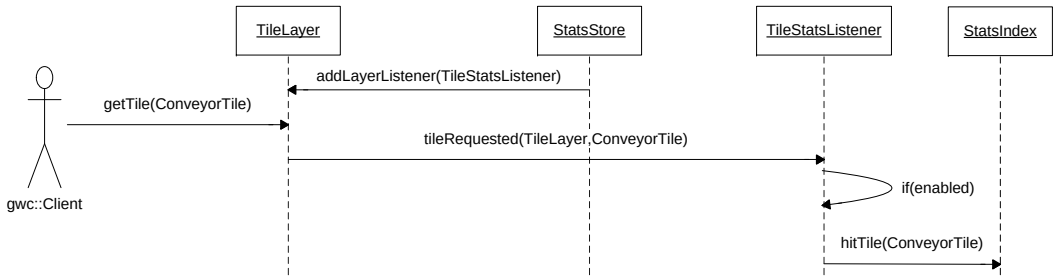


Figura A.28: Diagrama de secuencia para el almacenamiento de estadísticas.

a una fuente de fenómenos (*FeatureSource*). Éstas pueden obtenerse múltiples almacenes (*DataStore*), como *shapefiles*, ficheros GML (*Geography Markup Language*), bases de datos, servicios WFS y otros formatos.

Como se muestra en la Figura A.30, el catálogo de fenómenos consiste en un mapa que contiene las colecciones de fenómenos indexadas por nombre. La utilidad de este catálogo puede extenderse para la selección avanzada de zonas de interés para tareas de precarga, truncado, etc.

A.3.4. Módulo de precarga de teselas

El módulo `gwc-prefetch` dota a GeoWebCache de la funcionalidad de realizar una precarga de teselas en base al catálogo de fenómenos descrito en la Sección A.3.3 y a las

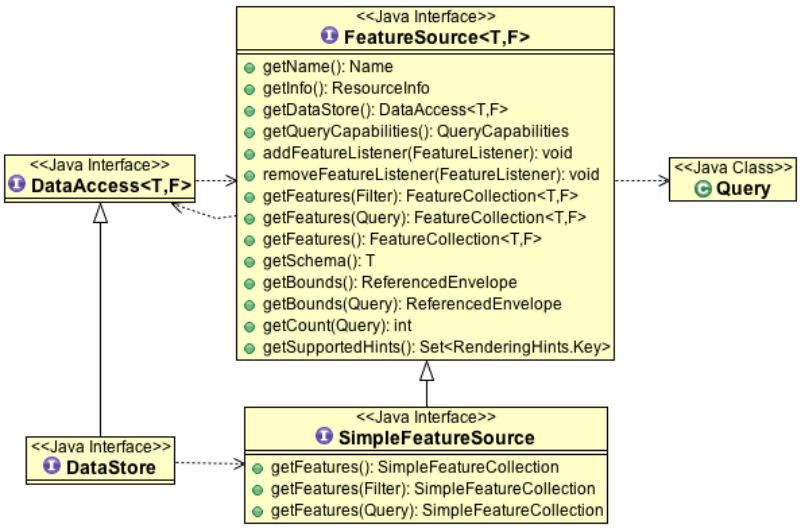


Figura A.29: Clases de GeoTools para el acceso a fuentes de fenómenos.

estadísticas almacenadas según se ha comentado en la Sección A.3.2.

Se ha desarrollado un componente `tilePrefetchingJobBean` en el que se configuran los parámetros de la tarea de precarga. La terna formada por el identificador de capa, rejilla y formato identifican la capa concreta que se desea cachear. Se puede seleccionar la zona geográfica a cachear (por defecto toda la capa) y el nivel de resolución deseado. Durante la navegación de los usuarios por el mapa es probable que no sólo se visiten las zonas atravesadas por los fenómenos de interés, sino también las zonas adyacentes. Por ello, se aplica un *buffer* configurable alrededor de las geometrías.

Tanto las estadísticas de las peticiones como los fenómenos del catálogo se rasterizan usando la rejilla definida a la escala a cachear. Para el rasterizado de las estadísticas se utiliza el contador de peticiones, mientras que para el de los fenómenos se puede seleccionar, o bien el valor de cualquier atributo asociado al fenómeno, o un valor binario (1/0) que indica la presencia/ausencia del fenómeno en cada tesela.

La rasterización de las estadísticas da lugar a una matriz B de tamaño $m \times n$, donde m y n son el número de teselas que abarca la zona a cachear, en los ejes horizontal y vertical, respectivamente. La rasterización de las l colecciones de fenómenos da lugar a l matrices A_i también de tamaño $m \times n$:

$$B_{[m\times n]} \qquad\qquad A_{1_{[m\times n]}} \cdots A_{l_{[m\times n]}}$$

$$\begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix} \qquad \begin{bmatrix} a_{1_{1,1}} & \cdots & a_{1_{1,n}} \\ \vdots & \ddots & \vdots \\ a_{1_{m,1}} & \cdots & a_{1_{m,n}} \end{bmatrix} \cdots \begin{bmatrix} a_{l_{1,1}} & \cdots & a_{l_{1,n}} \\ \vdots & \ddots & \vdots \\ a_{l_{m,1}} & \cdots & a_{l_{m,n}} \end{bmatrix}$$

Entonces, cada matriz $m \times n$ se transforma en un vector columna de longitud $m \cdot n$:

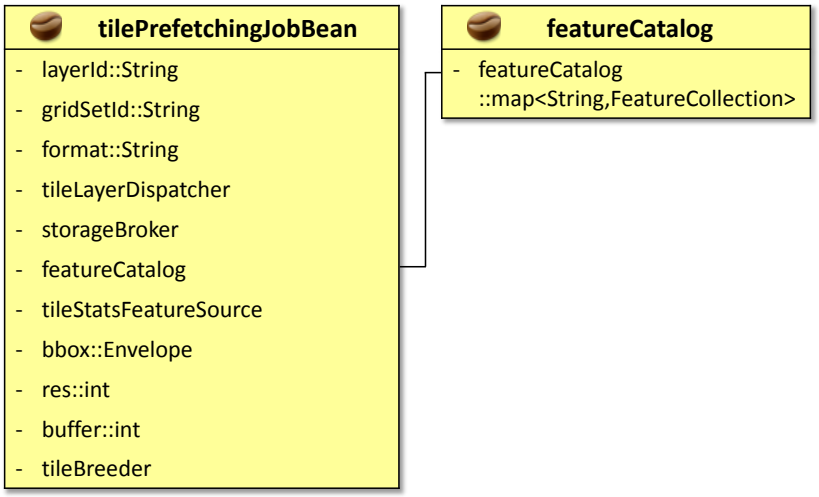


Figura A.30: Catálogo de fenómenos y componente de precarga.

$$Y_{[1,m\cdot n]} = \overline{B}_{[m\times n]} \qquad A'_{1_{[1\times(m\cdot n)]}} \cdots A'_{l_{[1\times(m\cdot n)]}}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{m\cdot n} \end{bmatrix} \qquad \begin{bmatrix} a'_{1_1} \\ \vdots \\ a'_{1_{m\cdot n}} \end{bmatrix} \cdots \begin{bmatrix} a'_{l_1} \\ \vdots \\ a'_{l_{m\cdot n}} \end{bmatrix}$$

Los vectores columna correspondientes a los fenómenos se concatenan horizontalmente para dar lugar a una matriz X de tamaño $(m \cdot n) \times l$:

$$X = [A'_{1_{[1\times(m\cdot n)]}} | A'_{2_{[1\times(m\cdot n)]}} \cdots | A'_{l_{[1\times(m\cdot n)]}}]$$

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,l} \\ \vdots & \ddots & \vdots \\ x_{m\cdot n,1} & \cdots & x_{m\cdot n,l} \end{bmatrix}$$

De esta forma ya podemos plantear una regresión lineal del tipo $Y = X^T\beta + \epsilon$, donde $\beta = [\beta_1, \cdots, \beta_{m\cdot n}]^T$ son los coeficientes de la regresión lineal y $\epsilon = [\epsilon_1, \cdots, \epsilon_{m\cdot n}]^T$ es el error que se pretende minimizar.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{m\cdot n} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,l} \\ \vdots & \ddots & \vdots \\ x_{m\cdot n,1} & \cdots & x_{m\cdot n,l} \end{bmatrix}^T \cdot \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{m\cdot n} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_{m\cdot n} \end{bmatrix} \tag{A.5}$$

La solución del problema OLS es el vector $\widehat{\beta}$ que minimiza el error cuadrático medio. Estos coeficientes se corresponden con los pesos de las distintas colecciones de fenómenos en la combinación lineal. Con estos coeficientes se calcula la salida del modelo $\widehat{Y} = X^T\widehat{\beta}$, que es un vector columna de longitud $m \cdot n$.

Este vector se re-dimensiona para dar lugar de nuevo a una matriz de tamaño $m \times n$, cuyos valores se mapean directamente a las $m \times n$ teselas de la zona de estudio. De esta forma, a cada tesela se le asigna un valor que debe ser interpretado como la estimación de accesos futuros a la misma. Esto nos permite establecer prioridades para la tarea de cacheo, por orden descendente en la estimación de accesos futuros.

La tarea de precarga se ha integrado utilizando el gestor de tareas de GeoWebCache. De esta forma, en cualquier momento puede consultarse el estado de completitud de las tareas a través de una interfaz REST.

Enviando una petición GET del tipo `/rest/seed/<layer name>.json` se obtiene una lista de las tareas pendientes ya planificadas y de las que se están ejecutando para la capa especificada. El contenido devuelto es un array en JSON (*JavaScript Object Notation*) como el siguiente:

```
{"long-array-array":[
  [<tiles processed>, <total # of tiles to process>, <remaining time>,
    <Task ID>, <Task status>],
  [<>, <>, <>, < >, < >],
  ...
]}.
```

Mediante una petición GET del tipo `/rest/seed.json` puede obtenerse información de todas las capas. Además, pueden eliminarse todas las tareas activas de una capa mediante una petición POST del tipo `/rest/seed/<layer name>`, o del tipo `/rest/seed` para terminar todas la tareas.

A.3.5. Posibles puntos de mejora

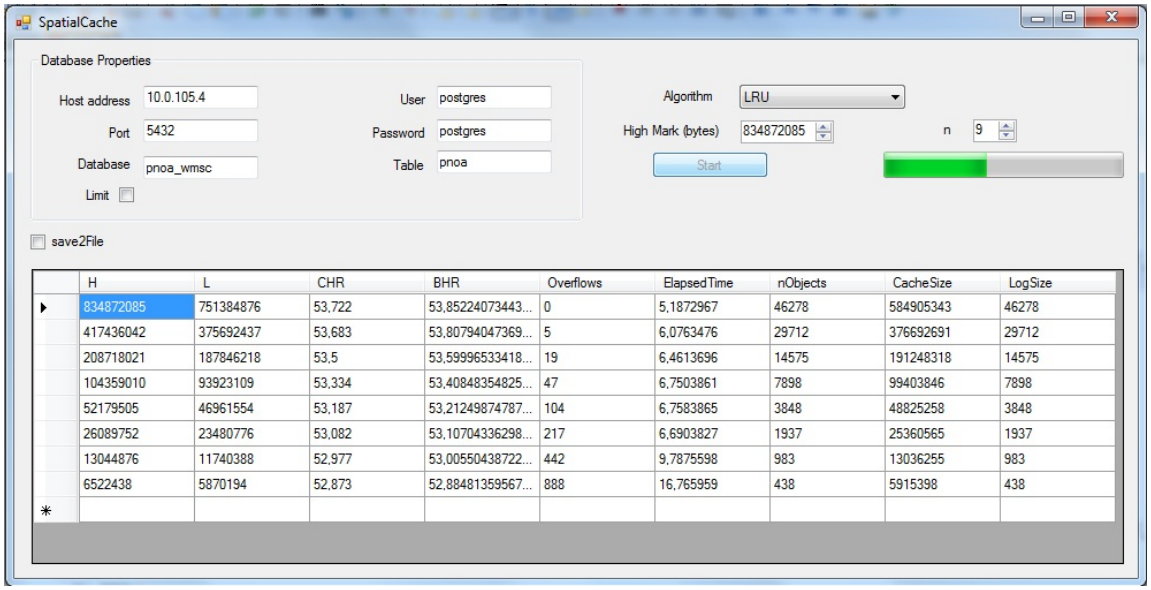
A modo de líneas futuras, se han detectado los siguientes puntos de mejora. Actualmente la tarea de precarga se lanza de forma programada a partir de un tiempo configurable después del arranque de la caché. Sería más adecuado que la tarea se ejecutase en función de la cantidad de estadísticas recogidas. En el futuro se desarrollará una interfaz Web amigable para la configuración de la tarea de precarga, como una alternativa a los ficheros de configuración actuales.

A.4. Simulador para la evaluación de políticas de reemplazo de una caché de mapas

Se ha desarrollado un simulador específico para la evaluación del rendimiento de las estrategias de reemplazo de caché propuestas. El uso de un simulador frente a una implementación real permite agilizar en gran medida realización de las pruebas.

Sobre este simulador se han implementado las siguientes políticas de gestión de caché: LRU, In-Cache LFU, Perfect-LFU, Spatial-LFU (ver Capítulo 6.2), Random, Max Size, Min Size, OPT, Worst y NNet (ver Capítulo 6.1).

El prototipo desarrollado simula el siguiente procedimiento, típico en la mayoría de implementaciones prácticas existentes (Podlipnig y Böszörmenyi, 2003): cuando se produce un fallo de caché, el objeto solicitado se almacena en la caché. Cuando el tamaño de ésta supera un límite pre-establecido, se elimina un cierto número de objetos para dejar espacio



H	L	CHR	BHR	Overflows	ElapsedTime	nObjects	CacheSize	LogSize
834872085	751384876	53,722	53,85224073443...	0	5,1872967	46278	584905343	46278
417436042	375692437	53,683	53,80794047369...	5	6,0763476	29712	376692691	29712
208718021	187846218	53,5	53,59996533418...	19	6,4613696	14575	191248318	14575
104359010	93923109	53,334	53,40848354825...	47	6,7503861	7898	99403846	7898
52179505	46961554	53,187	53,21249874787...	104	6,7583865	3848	48825258	3848
26089752	23480776	53,082	53,10704336298...	217	6,6903827	1937	25360565	1937
13044876	11740388	52,977	53,00550438722...	442	9,7875598	983	13036255	983
6522438	5870194	52,873	52,88481359567...	888	16,765959	438	5915398	438

Figura A.31: Captura del simulador de estrategias de reemplazo.

libre de cara a las peticiones entrantes. La caché utiliza dos marcas HM (*high mark*) y LM (*low mark*), con $HM > LM$. Cuando el tamaño de la caché excede HM, una política de reemplazo elimina objetos hasta que se alcanza el nivel LM.

El simulador utiliza como fuente de peticiones los datos almacenados en una base de datos PostgreSQL/PostGIS, poblada mediante los datos extraídos de los registros de los servidores de mapa, tal y como se describe en el Capítulo 5. A partir del encuadre geográfico de las peticiones se extraen los índices enteros $\{x, y, z\}$ que identifican la fila, columna y nivel de resolución de la tesela solicitada en la pirámide de escalas, respectivamente.

Dos peticiones de mapa corresponden al mismo objeto si ambas comparten los mismos valores para los siguientes parámetros de la petición WMS-C: *Layers*, *Format*, *SRS*, *Width*, *Height*, *Styles* y *BBox* (o de forma equivalente, x, y y z). Para acelerar el proceso de simulación, se crea un identificador único para cada objeto aplicando una función *hash* a los parámetros anteriores.

El simulador utiliza un *hashmap* para mantener los metadatos asociados a cada tesela almacenada en la caché, indexado mediante el identificador único ya comentado. Estos metadatos contienen la información necesaria para el funcionamiento de los algoritmos de reemplazo: tamaño del objeto, tiempo de último acceso, frecuencia de referencia, etc.

Algunos metadatos, como el contador de peticiones o el tiempo de último acceso, se actualizan por cada petición recibida. Cada política de reemplazo asigna una métrica diferente a los objetos utilizando estos metadatos, de forma que cuando es necesario liberar espacio de caché, los objetos con menor métrica son seleccionados para el reemplazo. Por ejemplo, en la estrategia LFU la métrica es el tiempo de último acceso.

A continuación se resume el modo de funcionamiento de este simulador:

1. Obtención de los registros de peticiones de la base de datos, ordenados temporalmente.
2. Creación de un objeto tesela (*tile*), con los siguientes parámetros: coordenadas $\{x, y, z\}$, tamaño de la imagen, tiempo de acceso y tiempo del próximo acceso (en caso de que

el objeto se pida de nuevo en el futuro, en caso contrario se asigna el valor ∞). El identificador de la tesela se construye como un *quadkey* con la terna $\{x, y, z\}$.

3. Se consulta si la tesela se encuentra almacenada en la caché (*hashmap*) utilizando su clave.

   a) Si la tesela se encuentra en caché:

      1) Se actualiza su contador y tiempo de último acceso.

   b) Si la tesela no se encuentra en caché:

      1) Se añade al *hashmap* con el contador inicializado a 1.
      2) Se añade al tamaño total de la caché el espacio que ocupa el nuevo objeto.
      3) Si el tamaño de la caché excede la marca HM se borran los objetos con menor métrica hasta que se alcanza LM, actualizando el tamaño total.

Este es el flujo general común a todas las estrategias de gestión implementadas. Sin embargo, algunas de ellas implementan alguna funcionalidad específica mediante pre-procesadores y post-procesadores en las etapas anteriores. Por ejemplo, en el caso del algoritmo Perfect-LFU, a parte del *hashmap* ya comentado que contiene los objetos que están actualmente en la caché, mantiene otro *hashmap* del cuál no se eliminan los objetos aunque sean borrados de la caché, de forma que la métrica de cada objeto presente en la caché se extrae de este otro mapa.

Apéndice B

Trabajos Relacionados con la Investigación

Resumen: Durante la realización del trabajo descrito en esta memoria se han realizado y obtenido diferentes resultados. En este anexo se recogen los trabajos realizados en el contexto de esta tesis, diferenciando los que están directamente relacionados con la temática de la memoria y los que no lo están.

B.1. Publicaciones derivadas de la investigación

B.1.1. Revista

Autores (p.o. de firma): Ricardo García Martín, Elena Verdú Pérez, Luisa María Regueras Santos, Juan Pablo de Castro Fernández, María Jesús Verdú Pérez
Título: A Neural Network Based Intelligent System for Tile Prefetching in Web Map Services
ISSN: 0957-4174
Revista: Expert Systems with Applications
Año: 2013
Índice de impacto JCR: 2.203
Journal Ranking:

- COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (22/111, Q1)
- ENGINEERING, ELECTRICAL & ELECTRONIC (41/245, Q1)
- OPERATIONS RESEARCH & MANAGEMENT SCIENCE (5/77, Q1)

Autores (p.o. de firma): Ricardo García Martín, Juan Pablo de Castro Fernández, Elena Verdú Pérez, María Jesús Verdú Pérez, Luisa María Regueras Santos
Título: An OLS Regression Model for Context-Aware Tile Prefetching in a Web Map Cache
ISSN: 1365-8816

Revista: International Journal of Geographical Information Science
Año: 2012
Índice de impacto JCR: 1.472
Journal Ranking:

- COMPUTER SCIENCE, INFORMATION SYSTEMS (36/133, Q2)
- GEOGRAPHY, PHYSICAL (23/43, Q3)

Autores (p.o. de firma): Ricardo García Martín, Juan Pablo de Castro Fernández, María Jesús Verdú Pérez, Elena Verdú Pérez, Luisa María Regueras Santos, Pablo López Escobés
Título: A Descriptive Model Based on the Mining of Web Map Server Logs for Tile Prefetching in a Web Map Cache
ISSN: 2074-1308
Revista: International Journal of Systems Applications, Engineering & Development
Año: 2011
Páginas: 469 - 476

B.1.2. Capítulos de libro

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras
Libro: Cartography - A Tool for Spatial Analysis.
ISBN: 978-953-51-0689-0.
Capítulo: Web Map Tile Services for Spatial Data Infrastructures: Management and Optimization
Páginas: inicial: 25, final: 48
Fecha: 2012
Editorial: InTech
Editor: Carlos Bateira
Lugar de publicación: Rijeka, Croatia

B.1.3. Congresos internacionales

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López
Título: A Cache Replacement Policy Based on Neural Networks Applied to Web Map Tile Caching
Congreso: The 2011 International Conference on Internet Computing, ICOMP 2011 (top 7,17 % del índice Computer Science Conference Ranking en el área Databases / Knowledge and Data Management / Data Security / Web / Mining) - WORLDCOMP 2011 (CORE C)
Publicación (ISSN/ISBN): Proceedings of the 2011 International Conference on Internet Computing, ICOMP 2011, CSREA Press, USA 2011, pp. 216-222. ISBN: 1-60132-186-4.

Lugar de Celebración: Las Vegas, Nevada, USA
Fecha: 18 - 21 Julio 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López
Título: An Adaptive Neural Network-Based Method for Tile Replacement in a Web Map Cache
Congreso: The 11th International Conference on Computational Science and Applications, ICCSA 2011 (CORE C, en el rank 3 del Computer Science Conference Ranking de la University of Alberta, top 66 % del índice CiteSEER))
Publicación (ISSN/ISBN): Computational Science and Its Applications - ICCSA 2011, LNCS 6782, B. Murgante et al. (Eds.), Springer-Verlag, Berlin Heidelberg 2011, pp. 76-91. ISBN: 978-3-642-21927-6.
Lugar de Celebración: University of Cantabria, Santander, Spain
Fecha: 20 - 23 Junio 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López
Título: A Descriptive Model for Predicting Popular Areas in a Web Map
Tipo de Participación: Comunicación
Congreso: The 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, AIKED 2011
Publicación (ISSN/ISBN): Proceedings of the 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, World Scientific and Engineering Academy and Society (WSEAS), 2011, pp. 397-402. ISBN: 978-960-474-273-8.
Lugar de Celebración: Cambridge, UK
Fecha: 20 - 22 Febrero 2011

B.1.4. Congresos nacionales

Autores (p.o. de firma): Pablo López, Ricardo García, Juan P. de Castro, María J. Verdú, Luisa M. Regueras, Elena Verdú
Título: Integración de APIs políglotas de mapas en Google Web Toolkit: IDELabMapstraction GWT
Tipo de Participación: Comunicación
Congreso: V Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las V Jornadas de SIG Libre (SIG Libre 2011). Girona, Spain, Mar 2011. SIGTE (Girona, 2011), ISBN: 978-84-694-1624-2.
Lugar de Celebración: Girona, España
Fecha: 23 - 25 Marzo 2011

Autores (p.o. de firma): Pablo López, Ricardo García, Juan P. de Castro, María J. Verdú, Luisa M. Regueras, Elena Verdú
Título: Definición e implementación de soluciones basadas en APIs universales para la integración de estándares OGC

Tipo de Participación: Comunicación
Congreso: V Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las V Jornadas de SIG Libre (SIG Libre 2011). Girona, Spain, Mar 2011. SIGTE (Girona, 2011), ISBN: 978-84-694-1624-2.
Lugar de Celebración: Girona, España
Fecha: 23 - 25 Marzo 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López
Título: WMSCWrapper: caché de teselas OpenSource para la aceleración de servicios de mapas teselados
Tipo de Participación: Comunicación
Congreso: V Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las V Jornadas de SIG Libre (SIG Libre 2011). Girona, Spain, Mar 2011. SIGTE (Girona, 2011), ISBN: 978-84-694-1624-2.
Lugar de Celebración: Girona, España
Fecha: 23 - 25 Marzo 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López
Título: Visualización geográfica 3D mediante un API Universal y políglota. Acercando los globos virtuales al desarrollador de mashups de mapas
Tipo de Participación: Comunicación
Congreso: V Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las V Jornadas de SIG Libre (SIG Libre 2011). Girona, Spain, Mar 2011. SIGTE (Girona, 2011), ISBN: 978-84-694-1624-2.
Lugar de Celebración: Girona, España
Fecha: 23 - 25 Marzo 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras
Título: Desarrollo de un sistema inteligente para la precarga automática de teselas en GeoWebCache a partir de un catálogo de fenómenos geográficos
Tipo de Participación: Comunicación
Congreso: III Jornadas Ibéricas de Infraestructuras de Datos Espaciales
Publicación (ISSN/ISBN): Actas de las III Jornadas Ibéricas de Infraestructuras de Datos Espaciales (JIIDE'2012), Madrid, España, 2012
Lugar de Celebración: Madrid, España
Fecha: 17 -19 Octubre 2012

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras, Pablo López, Daniel García
Título: Estrategias de metatiling para la aceleración de servicios de mapas teselados en las infraestructuras de datos espaciales
Tipo de Participación: Comunicación
Congreso: X Jornadas de Ingeniería Telemática
Publicación (ISSN/ISBN): Actas de las X Jornadas de Ingeniería Telemática (JITEL),

Santander, España, 2011
Lugar de Celebración: Santander, España
Fecha: 28 - 30 Septiembre 2011

Autores (p.o. de firma): Pablo López, Ricardo García, Juan P. de Castro, María J. Verdú, Elena Verdú, Luisa M. Regueras
Título: Utilización de datos geográficos auxiliares para la optimización de caches espaciales
Tipo de Participación: Comunicación
Congreso: X Jornadas de Ingeniería Telemática
Publicación (ISSN/ISBN): Actas de las X Jornadas de Ingeniería Telemática (JITEL), Santander, España, 2011
Lugar de Celebración: Santander, España
Fecha: 28 -30 Septiembre 2011

Autores (p.o. de firma): Ricardo García, Juan P. de Castro
Título: Aceleración de servicios de mapas teselados mediante la aplicación de un modelo descriptivo
Tipo de Participación: Comunicación
Congreso: I Jornadas Ibéricas de Infraestructuras de Datos Espaciales
Publicación (ISSN/ISBN): Actas de las I Jornadas Ibéricas de Infraestructuras de Datos Espaciales (JIIDE'2010), Lisbon, Portugal, 2010
Lugar de Celebración: Lisbon, Portugal
Fecha: 27 - 29 Octubre 2010

Autores (p.o. de firma): Pablo López, Juan P. de Castro, Ricardo García
Título: IDELab MapstractionInteractive: Api Universal y Políglota
Tipo de Participación: Comunicación
Congreso: IV Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las IV Jornadas de SIG Libre (SIG Libre 2010). Girona, Spain, Mar 2010. SIGTE (Girona, 2010)
Lugar de Celebración: Girona, España
Fecha: 10 -12 Marzo 2010

Autores (p.o. de firma): Ricardo García, Juan P. de Castro
Título: WMSCWrapper: Implementación WMS-C OpenSource para servicios WMS teselados.
Tipo de Participación: Comunicación
Congreso: IV Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las IV Jornadas de SIG Libre (SIG Libre 2010). Girona, Spain, Mar 2010. SIGTE (Girona, 2010)
Lugar de Celebración: Girona, España
Fecha: 10 -12 Marzo 2010

Autores (p.o. de firma): Ricardo García, Juan P. de Castro
Título: Benchmarking de implementaciones WMS-C de software libre
Tipo de Participación: Poster
Congreso: IV Jornadas de SIG libre

Publicación (ISSN/ISBN): Actas de las IV Jornadas de SIG Libre (SIG Libre 2010). Girona, Spain, Mar 2010. SIGTE (Girona, 2010)
Lugar de Celebración: Girona, España
Fecha: 10 -12 Marzo 2010

Autores (p.o. de firma): Ricardo García, Juan P. de Castro, Pablo López
Revista: IG+ Más que Información Geográfica
Título: API Universal y Políglota para la abstracción de clientes de mapas
Publicación (ISSN/ISBN): ISSN 1885-0715
Editorial: Servicio de SIG y Teledetección
Editor: Suansi Armisen
Lugar de publicación: Girona, España
Fecha: 2011

B.2. Publicaciones no directamente relacionadas

B.2.1. Revista

Autores (p.o. de firma): Elena Verdú Pérez, María Jesús Verdú Pérez, Luisa María Regueras Santos, Juan Pablo de Castro Fernández, Ricardo García Martín
Título: A genetic fuzzy expert system for automatic question classification in a competitive learning environment
ISSN: 0957-4174
Revista: Expert Systems with Applications
Año: 2012
Índice de impacto JCR: 2.203
Journal Ranking:

- COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (22/111, Q1)
- ENGINEERING, ELECTRICAL & ELECTRONIC (41/245, Q1)
- OPERATIONS RESEARCH & MANAGEMENT SCIENCE (5/77, Q1)

B.2.2. Congresos internacionales

Autores (p.o. de firma): Elena Verdú Pérez, María Jesús Verdú Pérez, Luisa María Regueras Santos, Juan Pablo de Castro Fernández, Ricardo García Martín
Título: Automatic Classification of Question Difficulty Level: Teachers´ Estimation vs. Students´ Perception
Tipo de Participación: Comunicación
Congreso: Frontiers in Education
Publicación (ISSN/ISBN): Actas del congreso *Frontiers in Education (FIE2012)*, Seattle (Washington), 2012

Lugar de Celebración: Seattle, Washington, EEUU
Fecha: 3 - 6 Octubre 2012

B.2.3. Congresos nacionales

Autores (p.o. de firma): Fernando Campos, Juan P. de Castro, Ricardo García
Título: Idelabroute: Librería para la gestión de grafos escalable
Tipo de Participación: Comunicación
Congreso: IV Jornadas de SIG libre
Publicación (ISSN/ISBN): Actas de las IV Jornadas de SIG Libre (SIG Libre 2010). Girona, Spain, Mar 2010. SIGTE (Girona, 2010)
Lugar de Celebración: Girona, España
Fecha: 10 -12 Marzo 2010

B.3. Participación en Proyectos de Investigación

Título del proyecto: eSalud: Control en movilidad

Entidad financiadora: Ministerio de Ciencia e Innovación a través del programa INNPACTO

Entidades participantes: Cotesa, Genasys, Fundación INTRAS, IDELab (Universidad de Valladolid)

Duración: desde Junio 2011 hasta Diciembre 2013 (Incorporación en Octubre de 2012)

Precio Total del Proyecto: 2.084.616 euros

Investigador responsable: Juan Pablo de Castro Fernández (U. Valladolid)

Título del proyecto: Convenio de Colaboración entre el Centro Nacional de Información Geográfica, del Ministerio de Fomento, y la Universidad de Valladolid para el desarrollo del proyecto "España Virtual" en el marco del programa CENIT del Centro para el Desarrollo Tecnológico Industrial del Ministerio de Ciencia e Innovación

Entidad financiadora: Ministerio de Ciencia e Innovación a través del Centro Nacional de Información Geográfica (CNIG)

Entidades participantes: Universidad de Valladolid

Duración: desde Enero 2009 hasta Diciembre 2011

Cuantía de la subvención: 779.218 euros

Investigador responsable: Juan Pablo de Castro Fernández (U. Valladolid)

Título del contrato/proyecto: Creación del aula confederación hidrográfica del Duero para el avance de las tecnologías espaciales en la gestión del agua.

Tipo de contrato: art. 83 - Investigación

Empresa/Administración financiadora: Confederación Hidrográfica del Duero

Entidades participantes: Universidad de Valladolid

Duración: desde Enero 2010 hasta Diciembre 2010

Investigador responsable: Juan Pablo de Castro Fernández

Número de investigadores participantes: 6

Precio Total del Proyecto: 67.640,00 euros

Título del contrato/proyecto: Investigación y desarrollo de servicios WMS-C, de aplicaciones cliente para la visualización y consulta de servicios de mapas, basadas en la integración de APIs, especialmente las llamadas APIs políglotas, y de servicios de geoprocesamiento

Tipo de contrato: art. 83 - Investigación

Empresa/Administración financiadora: Instituto Geográfico Nacional

Entidades participantes: Universidad de Valladolid

Duración: desde Enero 2009 hasta Enero 2010

Investigador responsable: Juan Pablo de Castro Fernández

Número de investigadores participantes: 6

Precio Total del Proyecto: 60.000 euros

B.4. Becas de investigación

Tipo de ayuda: Ayuda FPI-UVa para la Formación de Personal Investigador

Forma de acceso: Concurrencia competitiva en función de los méritos del candidato

Empresa/Administración financiadora: Universidad de Valladolid, cofinanciada por el Banco Santander

Duración: desde Enero de 2012 hasta Octubre de 2012

B.5. Dirección de Proyectos Fin de Carrera

- Pablo Navarrete Almarza (Febrero 2010). *"Benchmarking de servicios WMS-C."* Tutores: Ricardo García y Juan P. de Castro. Titulación en Ingeniero Técnico de Telecomunicación, Especialidad de Sistemas Electrónicos. Universidad de Valladolid.
- Alberto Casero de la Calle (Abril 2011). *"Implantación de un servicio WPS para la librería RouteEngine."* Tutores: Ricardo García y Juan P. de Castro. Titulación en Ingeniero Técnico de Telecomunicación, Especialidad de Sistemas Electrónicos. Universidad de Valladolid.
- Cristina Pérez Ortín (Junio 2011). *"Desarrollo de mecanismos para la población de una caché de mapas a partir de features vectoriales."* Tutores: Ricardo García y Juan P. de Castro. Titulación en Ingeniero Técnico de Telecomunicación, Especialidad en Telemática. Universidad de Valladolid.
- Pablo Caballero Escudero (Julio 2011). *"Mejora del rendimiento de la caché de teselas WMSCWrapper usando índices espaciales."* Tutores: Ricardo García y Juan P. de Castro. Titulación en Ingeniero Informático. Universidad de Valladolid.
- Jordana Torres González (Septiembre 2011). *"Visualización geográfica 3D mediante el API Universal y Políglota Mapstraction. Integración del globo virtual de la Nasa WorldWind en esta librería."* Tutores: Ricardo García y Juan P. de Castro. Titulación en Ingeniero de Telecomunicación. Universidad de Valladolid.

Bibliografía

Sede Electrónica del Catastro. 2013. Disponible en `http://www.sedecatastro.gob.es` (último acceso: Enero, 2013).

Abrams, M., Standridge, C. R., Abdulla, G., Fox, E. A. y Williams, S. Removal policies in network caches for World-Wide Web documents. *ACM SIGCOMM Computer Communication Review*, vol. 26(4), página 293–305, 1996.

Adrion, W. R. Research methodology in software enginering: summary of the dagstuhl workshop on future directions on software engineering. *SIGSoft Software Engineering Notes*, vol. 18(1), páginas 36–37, 1993.

Aggarwal, C., Wolf, J. y Yu, P. Caching on the World Wide Web. *Knowledge and Data Engineering, IEEE Transactions on*, vol. 11(1), páginas 94 –107, 1999. ISSN 1041-4347.

Aime, A. y McKenna, J. WMS Performance Shootout at FOSS4G. Informe técnico, OSGeo, Sydney, 2009.

Ali, W. y Shamsuddin, S. Neuro-fuzzy system in web client-side caching. En *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, páginas 888 –895. 2009.

Allan Doyle, editor. *OpenGIS Web Map Server Interface Implementation Specification - Revision 1.0.0*. Open Geospatial Consortium Inc, OGC 00-028, 2000. Disponible en `http://www.opengeospatial.org/standards/wms` (último acceso: Febrero, 2013).

Anselin, L. Local indicators of spatial association - LISA. *Geographical Analysis*, vol. 27, páginas 93–115, 1995.

Anselin, L., Syabri, I. y Kho, Y. GeoDa: An Introduction to Spatial Data Analysis. *Geographical Analysis*, vol. 38(1), páginas 5–22, 2006. ISSN 1538-4632.

Apache Software Foundation. Archivos de Registro (Log Files) del Servidor HTTP Apache. 2011. Disponible en `http://httpd.apache.org/docs/2.0/es/logs.html` (último acceso: Febrero, 2013).

Arlitt, M., Cherkasova, L., Dilley, J., Friedrich, R. y Jin, T. Evaluating content management techniques for Web proxy caches. *SIGMETRICS Perform. Eval. Rev.*, vol. 27, páginas 3–11, 2000. ISSN 0163-5999.

Arlitt, M. F. y Williamson, C. L. Web server workload characterization: the search for invariants. *SIGMETRICS Perform. Eval. Rev.*, vol. 24(1), páginas 126–137, 1996. ISSN 0163-5999.

ARLITT, M. F. y WILLIAMSON, C. L. Internet Web servers: workload characterization and performance implications. *IEEE/ACM Trans. Netw.*, vol. 5(5), páginas 631–645, 1997. ISSN 1063-6692.

BAHN, H., KOH, K., NOH, S. y LYUL, S. Efficient replacement of nonuniform objects in Web caches. *Computer*, vol. 35(6), páginas 65 –73, 2002. ISSN 0018-9162.

BARCLAY, T., GRAY, J. y SLUTZ, D. Microsoft TerraServer: a spatial data warehouse. En *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, páginas 307–318. ACM, New York, NY, USA, 2000. ISBN 1-58113-217-4.

DE LA BEAUJARDIERE, J., editor. *OpenGIS Web Map Server Implementation Specification v1.3.0. OpenGIS project document 06-042*. Open Geospatial Consortium Inc, 2006. Disponible en `http://www.opengeospatial.org/standards/wms` (último acceso: Diciembre, 2012).

BELL, D., KUEHNEL, F., MAXWELL, C., KIM, R., KASRAIE, K., GASKINS, T., HOGAN, P. y COUGHLAN, J. NASA World Wind: Opensource GIS for Mission Operations. En *Aerospace Conference, 2007 IEEE*, páginas 1 –9. 2007. ISSN 1095-323X.

BERGAMINI, J. A. y HAUNGS, D. M. Enabling P2P Cooperative WMS Proxy Caching and Prefetching in an Educational Environment. *The European Information Society*, páginas 1–14, 2007.

BERGAMINI, J. A. y HAUNGS, M. Geotorrent: Optimizing GIS web services for interactive educational use. *Proceedings of the UCGIS 2006 Summer Assembly*, 2006.

BLOWER, J. D. GIS in the cloud: implementing a web map service on Google App Engine. En *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, COM.Geo '10, páginas 34:1–34:4. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0031-5.

BRESENHAM, J. E. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, vol. 4(1), páginas 25 –30, 1965. ISSN 0018-8670.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G. y SHENKER, S. Web caching and Zipf-like distributions: evidence and implications . En *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, páginas 126 –134 vol.1. 1999. ISSN 0743-166X.

BUNZEL, K., AGER, A. y SCHRADER-PATTON, C. Up in the air: adventures in serving geospatial data using open source software and the cloud. En *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, COM.Geo '10, páginas 35:1–35:4. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0031-5.

BUTLER, P. Visualizing Facebook Friends: Eye Candy in R. 2010. Disponible en `http://paulbutler.org/archives/visualizing-facebook-friends/` (último acceso: Febrero, 2013).

CAMPOS, F., DE CASTRO, J. P. y GARCÍA, R. IDELabRoute: Librería para la gestión de grafos escalable. En *Actas de las IV Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2010.

CAO, P. y IRANI, S. Cost-aware WWW proxy caching algorithms. En *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, páginas 18–18. USENIX Association, Berkeley, CA, USA, 1997.

CHO, G. Using Predictive Prefetching to Improve Location Awareness of Mobile Information Service. En *Computational Science - ICCS 2002* (editado por P. Sloot, A. Hoekstra, C. Tan y J. Dongarra), vol. 2331 de *Lecture Notes in Computer Science*, páginas 1128–1136. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-47789-6_119.

CLAUSET, A., SHALIZI, C. R. y NEWMAN, M. E. J. Power-Law Distributions in Empirical Data. *SIAM Review*, vol. 51(4), páginas 661–703, 2009. ISSN 0036-1445, 1095-7200.

CLIFF, A. y ORD, J. *Spatial Processes: Models & Applications*, vol. 44. Pion London, 1981. ISBN 0850860814.

COBB, J. y ELAARAG, H. Web proxy cache replacement scheme based on back-propagation neural network. *Journal of Systems and Software*, vol. 81(9), páginas 1539–1558, 2008. ISSN 0164-1212.

COOLEY, R., MOBASHER, B. y SRIVASTAVA, J. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, vol. 1(1), páginas 5–32, 1999.

CUNHA, C., BESTAVROS, A. y CROVELLA, M. Characteristics of www client-based traces. Informe técnico, Boston, MA, USA, 1995.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, páginas 303–314, 1989. ISSN 0932-4194. 10.1007/BF02551274.

DAISEY, P., editor. *OpenGIS GeoPackage Implementation Specification. OpenGIS project document 12-128r1*. Open Geospatial Consortium Inc, 2012. Disponible en `http://www.opengeospatial.org/projects/groups/geopackageswg` (último acceso: Febrero, 2013).

DE SMITH, M., J, G., M F, L. y P, A. *Geospatial analysis: a Comprehensive Guide to Principles, Techniques and Software Tools*, vol. 1. Troubador, 2007.

DEMUTH, H., BEALE, M. y WORKS, M. *MATLAB: Neural Network Toolbox: User's Guide*. Math Works, 1992.

DENNING, P. J. The locality principle. *Commun. ACM*, vol. 48(7), páginas 19–24, 2005.

DEOLIVEIRA, J. GeoServer: uniting the GeoWeb and spatial data infrastructures. En *Proceedings of the 10th International Conference for Spatial Data Infrastructure, St. Augustine, Trinidad*. 2008.

DOYLE, A. y CUTHBERT, A. Essential Model of Interactive Portrayal. OpenGIS project document 98-061. 1998. Disponible en `http://www.opengis.org` (último acceso: Septiembre, 2012).

EL KHAYARI, R., OBAIDAT, M. y CELIK, S. An Adaptive Neural Network-Based Method for WWW Proxy Caches. *IAENG International Journal of Computer Science*, vol. 36(1), páginas 8–16, 2009. ISSN 1819-656X.

ELAARAG, H. y COBB, J. A Framework for using neural networks for web proxy cache replacement. *SIMULATION SERIES*, vol. 38(2), página 389, 2006. ISSN 0735-9276.

ELAARAG, H. y ROMANO, S. Improvement of the neural network proxy cache replacement strategy. En *Proceedings of the 2009 Spring Simulation Multiconference*, páginas 1–8. Society for Computer Simulation International, 2009a.

ELAARAG, H. y ROMANO, S. Training of NNPCR-2: An improved neural network proxy cache replacement strategy. En *Performance Evaluation of Computer Telecommunication Systems, 2009. SPECTS 2009. International Symposium on*, vol. 41, páginas 260 –267. 2009b.

ERLE, S., HOLMES, C., MARON, M. y WALSH, J. WMS-C WMS Tile Caching. 2006. Disponible en `http://wiki.osgeo.org/wiki/WMS_Tile_Caching` (último acceso: Febrero, 2013).

ÁNGEL ESBRÍ PALOMARES, M. y VALERO, J. V. H. Pruebas benchmark de soluciones cliente/servidor en software libre. En *Actas de las Jornadas Técnicas de la IDE de España JIDEE'05*. Madrid, 2005.

FANG, Y., JIANG, C. y FU, Y. Incremental Data Prefetching for Map Service in Mobile Navigation Application. En *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, páginas 1 –4. 2008.

FISHER, D. Hotmap: Looking at Geographic Attention. *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13(6), páginas 1184 –1191, 2007a. ISSN 1077-2626.

FISHER, D. How We Watch the City: Popularity and Online Maps. 2007b. Disponible en `http://research.microsoft.com/apps/pubs/default.aspx?id=69421` (último acceso: Septiembre, 2012).

FISHER, D. The Impact of Hotmap. 2009. Disponible en `http://research.microsoft.com/apps/pubs/default.aspx?id=81244` (último acceso: Septiembre, 2012).

FRÄNTI, P., AGEENKO, E., KOPYLOV, P., GRÖHN, S. y BERGER, F. Map image compression for real-time applications. En *Proc. Spatial Data Handling 2002 Symposium"SDH'2002",(part of 2002 Joint International Symposium on Geospatial Theory, Processing and Applications)*. 2002.

FRÄNTI, P., AGEENKO, E., KOPYLOV, P., GRÖHN, S. y BERGER, F. Compression of map images for real-time applications. *Image and Vision Computing*, vol. 22(13), páginas 1105 – 1115, 2004. ISSN 0262-8856.

GALLANT, S. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, vol. 1(2), páginas 179 –191, 1990. ISSN 1045-9227.

GARCÍA, R. y DE CASTRO, J. P. Aceleración de servicios de mapas teselados mediante la aplicación de un modelo descriptivo. En *Actas de las I Jornadas Ibéricas de Infraestructuras de Datos Espaciales (JIIDE'2010), Lisboa, Portugal*. 2010a.

GARCÍA, R. y DE CASTRO, J. P. Benchmarking de implementaciones WMS-C de software libre. En *Actas de las IV Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2010b.

García, R. y de Castro, J. P. WMSCWrapper. Implementación WMS-C OpenSource para servicios WMS teselados. En *Actas de las IV Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2010c.

García, R., de Castro, J. P. y López, P. *API Universal y Políglota para la abstracción de clientes de mapas*. Servicio de SIG y Teledetección, Girona, España, 2011a. Disponible en `www.sigte.udg.edu/sigte_es/revista_ig` (último acceso: Enero, 2013).

García, R., de Castro, J. P., López, P., Verdú, M. J., Regueras, L. M. y Verdú, E. WMSCWrapper: caché de teselas OpenSource para la aceleración de servicios de mapas teselados. En *Actas de las V Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2011b.

García, R., de Castro, J. P., Verdú, E., Verdú, M. J. y Regueras, L. M. An OLS Regression Model for Context-Aware Tile Prefetching in a Web Map Cache. *International Journal of Geographical Information Science*, 2012a. ISSN 1365-8816.

García, R., de Castro, J. P., Verdú, E., Verdú, M. J. y Regueras, L. M. Desarrollo de un sistema inteligente para la precarga automática de teselas en GeoWebCache a partir de un catálogo de fenómenos geográficos. En *III Jornadas Ibéricas de Infraestructura de Datos Espaciales*. Madrid, España, 2012b.

García, R., de Castro, J. P., Verdú, M. J., Verdú, E., Regueras, L. M. y López, P. A Cache Replacement Policy Based on Neural Networks Applied to Web Map Tile Caching. En *The 2011 International Conference on Internet Computing - (ICOMP)*. Las Vegas, Nevada, USA, 2011c.

García, R., de Castro, J. P., Verdú, M. J., Verdú, E., Regueras, L. M. y López, P. A Descriptive Model Based on the Mining of Web Map Server Logs for Tile Prefetching in a Web Map Cache. *International Journal of Systems Applications, Engineering & Development*, vol. 5(4), páginas 469–476, 2011d. ISSN 2074-1308.

García, R., de Castro, J. P., Verdú, M. J., Verdú, E., Regueras, L. M. y López, P. A descriptive model for predicting popular areas in a web map. En *Proceedings of the 10th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*, AIKED'11, páginas 397–402. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2011e. ISBN 978-960-474-273-8.

García, R., de Castro, J. P., Verdú, M. J., Verdú, E., Regueras, L. M. y López, P. An Adaptive Neural Network-Based Method for Tile Replacement in a Web Map Cache. En *Computational Science and Its Applications - ICCSA 2011* (editado por B. Murgante, O. Gervasi, A. Iglesias, D. Taniar y B. Apduhan), vol. 6782 de *Lecture Notes in Computer Science*, páginas 76–91. Springer Berlin / Heidelberg, 2011f. ISBN 978-3-642-21927-6. 10.1007/978-3-642-21928-3_6.

García, R., de Castro, J. P., Verdú, M. J., Verdú, E., Regueras, L. M., López, P. y García, D. Estrategias de Metatiling para la Aceleración de Servicios de Mapas Teselados en las Infraestructuras de Datos Espaciales. En *Actas de las X Jornadas de Ingeniería Telemática (JITEL), Santander, España*. Universidad de Cantabria, 2011g. ISBN 978-84-694-5948-5.

GARCÍA, R., TORRES, J., DE CASTRO, J. P., LÓPEZ, P., VERDÚ, M. J., REGUERAS, L. M. y VERDÚ, E. Visualización geográfica 3D mediante un API Universal y Políglota. Acercando los globos virtuales al desarrollador de mashups de mapas. En *Actas de las V Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2011h.

GARCÍA, R., VERDÚ, E., REGUERAS, L. M., DE CASTRO, J. P. y VERDÚ, M. J. A Neural Network Based Intelligent System for Tile Prefetching in Web Map Services. *Expert Systems with Applications*, 2013. ISSN 0957-4174.

GEARY, R. C. The contiguity ratio and statistical mapping. *The incorporated statistician*, vol. 5(3), páginas 115–146, 1954.

GERLEK, M. y FLEAGLE, M. Imaging on the Geospatial Web Using JPEG 2000. En *The Geospatial Web* (editado por A. Scharl y K. Tochtermann), Advanced Information and Knowledge Processing, páginas 27–38. Springer London, 2007. ISBN 978-1-84628-826-5.

GOODCHILD, M. Optimal tiling for large cartographic databases. En *Proceedings of AUTO-CARTO*, vol. 9, páginas 444–451. 1989.

GUAN, J., WANG, L. y ZHOU, S. Enabling GIS services in a P2P environment. En *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, páginas 776 – 781. 2004.

HAGAN, M. y MENHAJ, M. Training feedforward networks with the Marquardt algorithm. *Neural Networks, IEEE Transactions on*, vol. 5(6), páginas 989 –993, 1994. ISSN 1045-9227.

HASSANEIN, H., LIANG, Z. y MARTIN, P. Performance comparison of alternative web caching techniques. En *Proceedings of the Seventh International Symposium on Computer and Communications*, página 213–218. 2002.

HEFLEY, B., MURPHY, W., KARIMI, H. A. y ROONGPIBOONSOPIT, D. Are Clouds Ready for Geoprocessing? En *Cloud Computing and Services Science* (editado por I. Ivanov, M. van Sinderen y B. Shishkov), Service Science: Research and Innovations in the Service Economy, páginas 295–312. Springer New York, 2012. ISBN 978-1-4614-2326-3.

HONDA, K., HUNG, N. D. y SHIMAMURA, H. Linking ogc web services to google earth. En *SICE-ICASE, 2006. International Joint Conference*, páginas 4836 –4839. 2006.

HOPFIELD, J. y TANK, D. Neural computation of decisions in optimization problems. *Biological Cybernetics*, vol. 52, páginas 141–152, 1985. ISSN 0340-1200.

HU, C., ZHAO, Y., LI, J., MA, D. y LI, X. Geospatial Web Service for Remote Sensing Data Visualization. *Advanced Information Networking and Applications, International Conference on*, vol. 0, páginas 594–601, 2011. ISSN 1550-445X.

ISO. ISO/TC 211 Geographic Information/Geomatics: Standards Guide. Informe técnico, ISO/IEC, 2009.

KANG, V. K., PARK, Y. W., HWANG, B. y KIM, Y. S. Performance Profits of Tile Prefetching in Multi-level Abstracted Web Geographic Information Systems. En *Proceedings of the Int. Symposium on Internet and Multimedia*. Indonesia, 2002.

KANG, Y.-K., KIM, K.-C. y KIM, Y.-S. Probability-Based Tile Pre-fetching and Cache Replacement Algorithms for Web Geographical Information Systems. En *Proceedings of the 5th East European Conference on Advances in Databases and Information Systems*, ADBIS '01, páginas 127–140. Springer-Verlag, London, UK, 2001. ISBN 3-540-42555-1.

KÖBBEN, B. RIMapperWMS: a Web Map Service providing SVG maps with a built-in client. En *The European Information Society* (editado por S. Fabrikant y M. Wachowicz), Lecture Notes in Geoinformation and Cartography, páginas 217–230. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72384-4.

KEFALOUKOS, P., SALLES, M. y ZACHARIASEN, M. TileHeat: A Framework for Tile Selection. En *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2012)*. 2012.

KHAJOUEINEJAD, S., SABEGHI, M. y SADEGHZADEH, A. A fuzzy cache replacement policy and its experimental performance assessment. En *Innovations in Information Technology, 2006*, páginas 1 –5. 2006.

KHALID, H. A new cache replacement scheme based on backpropagation neural networks. *SIGARCH Comput. Archit. News*, vol. 25, páginas 27–33, 1997a. ISSN 0163-5964.

KHALID, H. Performance of the KORA-2 cache replacement scheme. *ACM SIGARCH Computer Architecture News*, vol. 25(4), páginas 17–21, 1997b. ISSN 0163-5964.

KHALID, H. y OBAIDAT, M. Application of neural networks to cache replacement. *Neural Computing & Applications*, vol. 8(3), páginas 246–256, 1999a. ISSN 0941-0643.

KHALID, H. y OBAIDAT, M. KORA-2: a new cache replacement policy and its performance. En *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, vol. 1, páginas 265 –269 vol.1. 1999b.

KIRCHNER, H., KRUMMENACHER, R., RISSE, T. y EDWARDS-MAY, D. A location-aware prefetching mechanism. En *Fourth International Network Conference (INC 2004)*, páginas 6–8. Citeseer, 2004.

KOCH, R. *The 80/20 Principle: The Secret to Achieving More with Less*. Crown Business, 1999. ISBN 0385491743.

KOTTMAN, C. y REED, C., editores. *The OpenGIS Abstract Specification Topic 5: Features v5.0.*. Open Geospatial Consortium Inc, OGC 08-126, 2009. Disponible en `http://www.opengeospatial.org/standards/as` (último acceso: Enero, 2013).

LANE, J. M., CARPENTER, L. C., WHITTED, T. y BLINN, J. F. Scan line methods for displaying parametrically defined surfaces. *Commun. ACM*, vol. 23(1), páginas 23–34, 1980. ISSN 0001-0782.

LEE, C.-H., CHEN, L., LEE, J.-D. y BAE, H.-Y. Content adaptation and transmission strategy of spatial information for WWW and mobile applications. En *Proceedings of the 2nd international conference on Human.society@internet*, HSI'03, páginas 12–22. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 3-540-40456-2.

LEE, D., KIM, J., KIM, S., KIM, K., YOO-SUNG, K. y PARK, J. Adaptation of a Neighbor Selection Markov Chain for Prefetching Tiled Web GIS Data. En *Advances in Information Systems* (editado por T. Yakhno), vol. 2457 de *Lecture Notes in Computer Science*, páginas 213–222. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-00009-9.

LI, R., GUO, R., XU, Z. y FENG, W. A prefetching model based on access popularity for geospatial data in a cluster-based caching system. *International Journal of Geographical Information Science*, 2012.

LIU, H. y NIE, Y. Tile-based Map Service GeoWebCache middleware. En *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 1, páginas 692 –697. 2010.

LIU, X., HAN, J., ZHONG, Y., HAN, C. y HE, X. Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS. En *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, páginas 1 –8. 2009. ISSN 1552-5244.

LOECHEL, A. J. y SCHMID, S. Caching techniques for high-performance Web Map Services. En *Proceedings of the AGILE'2012 International Conference on Geographic Information Science* (editado por D. J. Jérôme Gensel y D. Vandenbroucke), Lecture Notes in Geoinformation and Cartography, páginas 52–57. Springer-Verlag, Avignon, France, 2013. ISBN 978-90-816960-0-5.

LÓPEZ, P., DE CASTRO, J. P. y GARCÍA, R. Geo-habilitación de Gestores de Contenidos: CMSMap. En *Actas de las IV Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2010a.

LÓPEZ, P., DE CASTRO, J. P., GARCÍA, R., VERDÚ, M. J., REGUERAS, L. M. y VERDÚ, E. Definición e implementación de soluciones basadas en APIs Universales para la integración de estándares OGC. En *Actas de las V Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2011a.

LÓPEZ, P., GARCÍA, R. y DE CASTRO, J. P. IDELab MapstractionInteractive: API Universal y Políglota. En *Actas de las IV Jornadas de SIG Libre, Girona, España*. Universitat de Girona. Servei de Sistemes dÍnformació Geográfica i Teledetecció, 2010b.

LÓPEZ, P., GARCÍA, R., DE CASTRO, J. P., VERDÚ, M. J., REGUERAS, L. M. y VERDÚ, E. Utilización de datos geográficos auxiliares para la optimización de caches espaciales. En *Actas de las X Jornadas de Ingeniería Telemática (JITEL), Santander, España*. Universidad de Cantabria, 2011b. ISBN 978-84-694-5948-5.

MAHEMOFF, M. *Ajax Design Patterns*. O'Reilly Media, Inc., 2006. ISBN 0596101805.

MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, vol. 11(2), páginas 431–441, 1963.

MARTINEZ-LLARIO, J. y GONZALEZ-ALCAIDE, M. Design of a java spatial extension for relational databases. *Journal of Systems and Software*, vol. 84(12), páginas 2314 – 2323, 2011. ISSN 0164-1212.

MASÓ, J., JULIÀ, N. y PONS, X. Historia y estado actual del futuro estándar Web Map Tiling Service del OGC. En *Actas de las V Jornadas Técnicas de la Infraestructura de Datos Espaciales de España (JIDEE 2008)*. Tenerife (España), 2008.

MASÓ, J., POMAKIS, K. y JULIÁ, N. OpenGIS Web Map Tile Service Implementation Standard. OpenGIS project document 07-057r7. 2010. Disponible en `http://www.opengeospatial.org/standards/wmts` (último acceso: Septiembre, 2012).

MASSEY, F. J. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, vol. 46(253), páginas 68–78, 1951.

MCCLENDON, B., ROHLF, J. ET AL. Network link for providing dynamic data layer in a geographic information system. 2011. US Patent 7,933,929.

METACARTA. TileCache, from MetaCarta Labs. 2008. Disponible en `http://tilecache.org/` (último acceso: Febrero, 2013).

MITCHELL, T. M. *Machine Learning*. McGraw-Hill, New York, 1997.

MORTON, G. A computer oriented geodetic data base and a new technique in file sequencing. Informe Técnico Ottawa, Ontario, Canada, International Business Machines Company, 1966.

NEBERT, D., WHITESIDE, A. y VRETANOS, P., editores. *OpenGIS Catalogue Services Specification*. Open Geospatial Consortium Inc, 2007. Disponible en `http://www.opengeospatial.org/standards/cat` (último acceso: Diciembre, 2012). Published: OGC Implementation Specification.

OBAIDAT, M. y KHALID, H. Estimating neural networks-based algorithm for adaptive cache replacement. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 28(4), páginas 602 –611, 1998. ISSN 1083-4419.

OPENGEO. GeoWebCache. 2008. Disponible en `http://geowebcache.org` (último acceso: Diciembre, 2012).

OSGEO. Tile Map Service Specification. 2008a. Disponible en `http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification` (último acceso: Septiembre, 2012).

OSGEO. WMS Tiling Client Recommendation - OSGeo Wiki. 2008b. Disponible en `http://wiki.osgeo.org/wiki/WMS_Tiling_Client_Recommendation` (último acceso: Diciembre, 2012).

P., D. D., RODRIGO, J. y ROSALES, J. Medición de rendimientos de servicios WMS con JMeter. En *IDE, aplicaciones al planeamiento y la gestión del territorio*. Tenerife, 2008.

PARK, D.-J. y KIM, H.-J. Prefetch policies for large objects in a Web-enabled GIS application. *Data & Knowledge Engineering*, vol. 37(1), páginas 65 – 84, 2001. ISSN 0169-023X.

PENG, Z. R. y TSOU, M. H. *Internet GIS: distributed geographic information services for the internet and wireless networks*. John Wiley & Sons Inc, 2003.

PHIL YANG, C., WONG, D. W., YANG, R., KAFATOS, M. y LI, Q. Performance-improving techniques in web-based GIS. *International Journal of Geographical Information Science*, vol. 19(3), páginas 319–342, 2005.

PLESEA, L. The Design, Implementation and operation of the JPL OnEarth WMS Server. En *Geospatial Services and Applications for the Internet* (editado por J. Sample, K. Shaw, S. Tu y M. Abdelguerfi), páginas 93–109. Springer, Berlin, 2008. ISBN 978-0-387-74673-9.

PLESEA, L. OnEarth, JPL WMS Server. 2012. Disponible en `http://wms.jpl.nasa.gov` (último acceso: Enero, 2013).

PLEWE, B. *GIS Online: Information Retrieval, Mapping, and the Internet*. OnWord Press, 1st edición, 1997. ISBN 1566901375.

PODLIPNIG, S. y BÖSZÖRMENYI, L. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, vol. 35(4), páginas 374–398, 2003. ISSN 0360-0300.

POESE, I., UHLIG, S., KAAFAR, M. A., DONNET, B. y GUEYE, B. IP geolocation databases: unreliable? *SIGCOMM Comput. Commun. Rev.*, vol. 41(2), páginas 53–56, 2011. ISSN 0146-4833.

POMERENE, J., PUZAK, T. R., RECHTSCHAFFEN, R. y SPARACIO, F. Prefetching Mechanism for a high-speed buffer store. *US patent*, 1984.

PŘIDAL, K. P. Tiles à la Google Maps: Coordinates, Tile Bounds and Projection - conversion to EPSG:900913 (EPSG:3785) and EPSG:4326 (WGS84). 2008. Disponible en `http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/` (último acceso: Diciembre, 2012).

QUINN, S. y GAHEGAN, M. A Predictive Model for Frequently Viewed Tiles in a Web Map. *T. GIS*, vol. 14(2), páginas 193–216, 2010.

RIGAUX, P., SCHOLL, M. y VOISARD, A. *Spatial Database: With Application to Gis*. The Morgan Kaufmann Series in Data Management Systems Series. Morgan Kaufmann, 2002. ISBN 9781558605886.

ROMANO, S. y ELAARAG, H. A neural network proxy cache replacement strategy and its implementation in the Squid proxy server. *Neural Computing & Applications*, vol. 20(1), páginas 59–78, 2011. ISSN 0941-0643.

RUI, J. y GUI-LI, L. SOP4GIS: A New GIS Implement for Public Participation. En *Computer Science and Information Engineering, 2009 WRI World Congress on*, vol. 2, páginas 374 –378. 2009.

RUSHFORTH, P., editor. *OGC Canadian Geospatial Data Infrastructure WFS and GML Best Practices*. Open Geospatial Consortium Inc, OGC 08-002, 2007.

SCHWARTZ, J. Bing Maps Tile System. Microsoft Developer network, 2009. Disponible en `http://msdn.microsoft.com/en-us/library/bb259689.aspx` (último acceso: Diciembre, 2012).

SEBER, G. A. F. *Multivariate Distributions*, páginas 17–58. John Wiley & Sons, Inc., 1984. ISBN 9780470316641.

SHAN, T. C. y HUA, W. W. Data Caching Patterns. 2009.

SHEKHAR, S. y XIONG, H., editores. *Encyclopedia of GIS*. Springer, 2008. ISBN 978-0-387-30858-6.

SKYLAB MOBILESYSTEMS LTD. List of public OGC Web Services. 2005. Disponible en `http://www.skylab-mobilesystems.com/en/wms_serverlist.html` (último acceso: Julio, 2012).

SMITH, T. y LAKSHMANAN, V. Utilizing Google Earth as a GIS platform for weather applications. En *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*. 2006.

STALLINGS, W. *Computer Organization and Architecture - Designing for Performance (7. ed.)*. Pearson / Prentice Hall, 2006. ISBN 978-0-13-185644-8.

TALAGALA, N., ASAMI, S., PATTERSON, D., FUTERNICK, B. y HART, D. The art of massive storage: a Web image archive. *Computer*, vol. 33(11), páginas 22 – 28, 2000. ISSN 0018-9162.

TANENBAUM, A. S. *Modern Operating Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. ISBN 0130313580.

TIAN, W., CHOI, B. y PHOHA, V. An Adaptive Web Cache Access Predictor Using Neural Network. *Developments in Applied Artificial Intelligence*, páginas 113–117, 2002.

TOBLER, W. A computer movie simulating urban growth in the Detroit region. *Economic geography*, vol. 46, páginas 234–240, 1970. ISSN 0013-0095.

TOMLINSON, R. F., MARBLE, D. F. y CALKINS, H. W. *Computer Handling of Geographical Data: An Examination of Selected Geographic Information Systems*. Bernan Associates, 1979. ISBN 9231013408.

TU, S., HE, X., LI, X. y RATCLIFF, J. J. A systematic approach to reduction of user-perceived response time for GIS web services. En *Proceedings of the 9th ACM international symposium on Advances in geographic information systems*, GIS '01, páginas 47–52. ACM, New York, NY, USA, 2001. ISBN 1-58113-443-6.

VAKALI, A. Evolutionary techniques for web caching. *Distributed and Parallel Databases*, vol. 11, páginas 93–116, 2002. ISSN 0926-8782.

VENKATARAMANI, A., YALAGANDULA, P., KOKKU, R., SHARIF, S. y DAHLIN, M. The potential costs and benefits of long-term prefetching for content distribution. *Computer Communications*, vol. 25(4), páginas 367 – 375, 2002. ISSN 0140-3664.

VENKETESH, P. y VENKATESAN, R. A Survey on Applications of Neural Networks and Evolutionary Techniques in Web Caching. *IETE Technical Review*, vol. 26(3), páginas 171–180, 2009.

VERDÚ, E., VERDÚ, M. J., REGUERAS, L. M., DE CASTRO, J. P. y GARCÍA, R. A genetic fuzzy expert system for automatic question classification in a competitive learning environment. *Expert Systems with Applications*, vol. 39(8), páginas 7471 – 7478, 2012a. ISSN 0957-4174.

VERDÚ, E., VERDÚ, M. J., REGUERAS, L. M., DE CASTRO, J. P. y GARCÍA, R. Automatic Classification of Question Difficulty Level: Teachers´ Estimation vs. Students´ Perception. En *The 42nd IEEE international conference on Frontiers in Education conference (FIE2012)*. IEEE Press, 2012b.

VRETANOS, P. A., editor. *Web Feature Service Implementation Specification*. Open Geospatial Consortium Inc, OGC 04-094, 2005. Disponible en `http://www.opengeospatial.org/standards/wfs` (último acceso: Septiembre, 2012).

WANG, H., PAN, S., PENG, M. y LI, R. Zipf-like distribution and its application analysis for image data tile request in digital Earth. *Geomatics and Information Science of Wuhan University*, vol. 33(11), páginas 356–359, 2010.

WANG, W., ZHANG, Y., LI, Y. y ZHANG, X. The Global Fuzzy C-Means Clustering Algorithm. En *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1, páginas 3604 –3607. 2006.

WANG, Y., WANG, S. y ZHOU, D. Retrieving and Indexing Spatial Data in the Cloud Computing Environment. En *Cloud Computing* (editado por M. Jaatun, G. Zhao y C. Rong), vol. 5931 de *Lecture Notes in Computer Science*, páginas 322–331. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-10664-4. 10.1007/978-3-642-10665-1_29.

WEI, Z.-K., OH, Y.-H., LEE, J.-D., KIM, J.-H., PARK, D.-S., LEE, Y.-G. y BAE, H.-Y. Efficient spatial data transmission in Web-based GIS. En *Proceedings of the 2nd international workshop on Web information and data management*, WIDM '99, páginas 38–42. ACM, New York, NY, USA, 1999. ISBN 1-58113-221-2.

WERNECKE, J. *The KML Handbook: Geographic Visualization for the Web*. Addison-Wesley Professional, 1 edición, 2008. ISBN 0321525590.

WESSELS, D., NORDSTRÖM, H. y ROUSSKOV, A. squid : Optimising Web Delivery. http://www.squid-cache.org/, 2009.

WHITESIDE, A. y EVANS, J. D., editores. *Web Coverage Service (WCS) Implementation Standard. OpenGIS project document 07-067r5*. Open Geospatial Consortium, 2008. Disponible en `http://www.opengeospatial.org/standards/wcs` (último acceso: Septiembre, 2012).

XIAOQIANG, Y. y YUEJIN, D. Exploration of cloud computing technologies for geographic information services. En *Geoinformatics, 2010 18th International Conference on*, páginas 1 –5. 2010.

YANG, L., LIU, Y., XIA, Y. y LIU, W. Mining Measurement of Spatial Autocorrelation Based on Spatial Adjacency Index: A Case Study in Land Use of Jiangxia District in Wuhan City. En *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, páginas 4700–4703. 2008.

YESILMURAT, S. y ISLER, V. Retrospective adaptive prefetching for interactive Web GIS applications. *GeoInformatica*, vol. 16(3), páginas 435–466, 2012. ISSN 1384-6175. 10.1007/s10707-011-0141-8.

ZHANG, J. *Spatial trend prefetching for online maps mashups*. Proyecto Fin de Carrera, UBCV, University of British Columbia, 2008.

ZHANG, X., LI, G. y LAN, X. Research on WebGIS Performance Optimization. En *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, páginas 1 –4. 2011. ISSN 2161-9646.

ZIPF, G. K. *Human Behavior and the Principle of Least Effort*. Addison-Wesley (Reading MA), 1949.

Lista de acrónimos

AJAX *Asynchronous JavaScript and XML* 20
ANN *Artificial Neural Network*
API *Application Programming Interface* 21
BHR *Byte-Hit Ratio* 111
BLOB *Binary Large OBject* 32
CCR *Correct Classification Ratio* 74
CCDF *Complementary Cumulative Distribution Function*
CDF *Cumulative Distribution Function*
CDTI Centro para el Desarrollo Tecnológico Industrial VII
CHR *Cache-Hit Ratio* 111
CLF Formato Común de Registro - *Common Log Format* 44
CM *Cache Manager*
CMS *Content Management System* 21
CNIG Centro Nacional de Información Geográfica VII
CoV *Coefficient of Variation* 52
CRS *Coordinate Reference System* 18
CSW *Catalog Service for the Web* 88
DFS *Distributed File System* 34
DGT Dirección General de Tráfico 86
ETSIT Escuela Técnica Superior de Ingenieros de Telecomunicación 12
GIF *Graphics Interchange Format* 10
GIS *Geographic Information System*
GML *Geography Markup Language* 166
GWR *Geographically Weighted Regression* 132
GWT *Google Web Toolkit* 21
HDFS *Hadoop Distributed File System* 35
IDE Infraestructura de Datos Espacial 88
IDEE Infraestructuras de Datos Espaciales de España 2
IDEs Infraestructuras de Datos Espaciales 31

IGN Instituto Geográfico Nacional ... VII
ISP *Internet Service Provider* ... 53
JPEG *Joint Photographic Experts Group* ... 10
JPL *Jet Propulsion Laboratory* ... 9
JSON *JavaScript Object Notation* ... 169
KML *Keyhole Markup Language* ... 137
KVP *Key Value Pair* ... 16
LFU *Least Frequently Used* ... 7
LISA *Local Indicators of Spatial Association*
LOD *Level of Detail*
LRU *Least Recently Used*
MLE *Maximum Likelihood Estimator*
MLP *Multi-Layer Perceptron* ... 113
MLR *Multiple Linear Regression* ... 118
MSE *Minimum Square Error*
NSMC *Neighbor Selection Markov Chain* ... 37
OGC *Open Geospatial Consortium* ... 1
OLS *Ordinary Least Squares* ... IX
P2P *Peer to Peer* ... 33
PA *Prefetch Agent* ... 39
PE *Prefetch Executor* ... 38
PCA *Principal Components Analysis* ... 89
PNG *Portable Network Graphics* ... 10
PNOA Plan Nacional de Ortofotografía Aérea ... 12
PPGIS *Public Participatory Geographic Information System* ... 35
QAE *Query Analyzer and Executor* ... 38
QoS *Quality of Service* ... XI
REST *Representational State Transfer* ... 140
ROI *Region Of Interest*
SE *Search Engine*
SIG Sistemas de Información Geográfica ... 9
SLD *Styled Layers Descriptor* ... 13
SOA *Service-Oriented Architecture* ... 35
SOAP *Simple Object Access Protocol*
SVG *Scalable Vector Graphics* ... 10
SWL *Sliding Window Length* ... 72

TMS *Tiled Map Service*
URL *Uniform Resource Locator* 12
UVA Universidad de Valladolid 12
WCS *Web Coverage Service* 88
WebCGM *Web Computer Graphics Metafile* 10
WFS *Web Feature Service* 88
WKB *Well Known Binary* 34
WKT *Well Known Text* 34
WLS *Weighted Least Squares* 132
WMS *Web Map Service* 1
WMTS *Web Map Tile Service* 16
WMS-C *Web Map Service - Cached* 1