Utilización de estructuras compactas para la generalización cartográfica de información geográfica vectorial *

Nieves R. Brisaboa, Alejandro Cortiñas, Pablo Gutiérrez-Asorey, Miguel R. Luaces, Tirso V. Rodeiro

Universidade da Coruña, CITIC. Laboratorio de Bases de Datos
Facultade de Informática, Campus de Elviña s/n, 15071, A Coruña, España
{nieves.brisaboa,alejandro.cortinas,pablo.gutierrez,
luaces,tirso.varela.rodeiro}@udc.es

Resumen A pesar de la madurez tecnológica de los Sistemas de Información Geográfica, todavía existen retos de investigación relevantes en el campo relacionados con las problemáticas del almacenamiento y representación eficientes de la información. En este trabajo presentamos un proyecto de investigación en curso centrado en el desarrollo de nuevas estructuras de datos para almacenar información geográfica para contextos en los que se espera que se genere una cantidad de datos inmensa.

Keywords: Geoinformática · Estructuras de datos · Optimización

1. Motivación y retos

Comparando un trabajo de principios de los 2000 [1] con otro más reciente [2], podemos observar que ambos proponen una arquitectura cliente-servidor similar para construir una aplicación web basada en Sistemas de Información Geográfica (SIG). Esto es un síntoma del alto grado de madurez que estas tecnologías han alcanzado. Gracias a la estandarización de las soluciones basadas en SIG, hoy en día existe una amplia oferta de software que abarca todos los componentes de un sistema completamente funcional. A pesar de todo, todavía quedan líneas de investigación activas relacionadas con la optimización de estos servicios.

Debido a la popularidad de los dispositivos que trabajan con información geográfica, están surgiendo nuevas posibilidades en distintos contextos que pueden ser atendidas mediante la explotación de dicha información. Algunos ejemplos incluyen localización de flotas en mar abierto, análisis de movimientos migratorios de aves o localización en tiempo real de trabajadores. La creciente cantidad de información conlleva la necesidad de manejar grandes cantidades de

* Este trabajo ha sido parcialmente financiado por el CITIC (programa Xunta/FEDER-UE 2014-2020 [ED431G 2019/01]), MICIU (PGE/ERDF) [MaGIST: PID2019-105221RB-C41; Datos 4.0: TIN2016-78011-C4-1-R]; MICINN (PGE/ERDF) [EXTRA-Compact: PID2020-114635RB-I00], y por el programa FPI [BES-2017-081390]

información geográfica, que no siempre se lleva a cabo de la forma más eficiente, ni a la hora de almacenarla ni, especialmente, a la hora de transferirla.

Usando estructuras compactas (soluciones optimizadas en tamaño en las que la información sigue siendo fácilmente accesible) podemos solucionar los problemas de almacenamiento. Si además garantizamos que en las comunicaciones cliente-servidor solo es necesario transmitir la cantidad mínima de información, se agilizarán los procesos de transferencia de geometrías a la aplicación cliente.

Uno de los modelos de datos más comunes para almacenar la información geográfica es el modelo vectorial. Este modelo utiliza tipos de datos de uso común (por ejemplo, números enteros o en punto flotante) para definir un sistema de coordenadas sobre el que se representa la información geográfica utilizando construcciones geométricas como puntos (p. ej. posiciones de un elemento), polilíneas (p. ej. bordes de carreteras) y polígonos cerrados (p. ej. regiones). Para visualizar estas geometrías en aplicaciones web hay dos alternativas, o bien se crean imágenes en base a las mismas en el servidor y son estas imágenes lo que se envía al cliente, o el servidor envía los objetos completos al cliente, delegando la carga de procesamiento necesaria para la visualización en el mismo.

La primera de las alternativas es mucho más eficiente pero limita mucho las posibilidades de interacción con el mapa por parte del usuario. Debido a esto, la segunda alternativa está cobrando mucha popularidad, y es en la que centraremos nuestra investigación. El envío de los objetos geográficos completos al cliente tiene, no obstante, dos grandes problemas: por un lado, los objetos geográficos son pesados en tamaño; por otro lado, renderizar objetos geográficos en el cliente es una operación costosa, especialmente cuando las geometrías tienen un alto nivel de detalle, o cuando el número de objetos a visualizar es grande.

Una solución a estos problemas es aplicar generalización cartográfica, que consiste en almacenar diferentes versiones de las geometrías complejas en función de la granularidad requerida. Es decir, dependiendo de la escala (zoom) que el usuario esté usando en cada momento, el servidor enviará una versión más o menos simplificada de la geometría. Por supuesto, almacenar diferentes geometrías para el mismo objeto geográfico supone que una parte importante de la información que se está guardando es redundante (la versión con más detalle de una geometría contiene siempre a la versión de menos detalle, por ejemplo). En este trabajo introducimos una alternativa eficiente atacando este problema.

2. Nuestra propuesta

Entendemos generalización cartográfica como un proceso que permite transformar una representación de un cierto nivel de detalle alto a otro más genérico. En concreto, este trabajo se basa en la idea de almacenar versiones más sencillas de un objeto geográfico completo reduciendo el número de puntos que lo forman usando el algoritmo de Douglas-Peucker [3]. Este algoritmo toma una curva compuesta por segmentos lineales y la simplifica a otra con menos puntos.

En la Figura 1 ilustramos un ejemplo de cómo sería este proceso. Partimos del objeto **a)**, que se corresponde con el máximo nivel de detalle (10 puntos), y

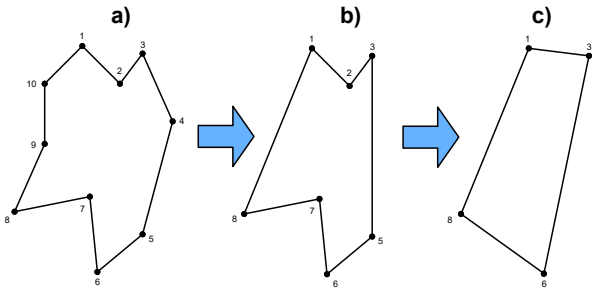

Figura 1. Ejemplo de generalización cartográfica con tres niveles de detalle, con a) el máximo nivel de detalle, b) el intermedio y c) el mínimo.

lo simplificamos primero a **b)** (7 puntos) y luego a **c)** (4 puntos). Estas nuevas representaciones podrían ser utilizadas en diferentes niveles de zoom en el cliente para representar únicamente el detalle visible en cada momento.

Nuestra propuesta parte de la idea de evitar guardar múltiples versiones de la misma geometría con los mismos puntos geográficos con el objetivo de conseguir una representación más compacta. En la Figura 2 presentamos una visión general de nuestra propuesta en base a los objetos de la Figura 1.

En la práctica, el proceso que seguimos parte de la aplicación consecutiva del algoritmo de Douglas-Peucker sobre el objeto geométrico a almacenar hasta conseguir los niveles de detalle deseados (3 en el caso del ejemplo). Por lo tanto, el nivel de mínimo en este caso se corresponde con el objeto **c)** en la Figura 1. A partir de este primer nivel, todo nivel sucesivo contiene solo los puntos adicionales necesarios para su representación. Esto significa que la representación de cada nivel es la unión de sus puntos con los de todos los niveles superiores de detalle y, a su vez, la unión de todos los niveles de detalle (zoom máximo) contiene todos los puntos del objeto original. De esta forma, nuestra propuesta nunca almacena el mismo punto más de una vez. Finalmente, para cada nivel de la representación definimos un bitmap (B1, B2, B3 en la Figura 2) en el que marcamos a 1 los puntos incluidos en dicho nivel de detalle.

Partiendo de esta base, planteamos que es posible reducir el tamaño que las geometrías ocupan en memoria utilizando una codificación diferencial [4] para los valores de coordenadas en los puntos. El objetivo es guardar números más pequeños que requieren menos bits en memoria. Para esto aprovecharemos el principio de localidad espacial que determina qué coordenadas próximas en la secuencia estarán próximas espacialmente. Esto implica que la diferencia entre los valores de las coordenadas será pequeña y que por tanto será más eficiente almacenar la diferencia entre las coordenadas que las propias coordenadas.

Siguiendo este razonamiento, nuestra idea es que solo el nivel mínimo de zoom contenga valores de coordenadas reales, y cada nivel sucesivo utilice valores relativos al primer valor en el nivel inmediatamente superior a su izquierda. Es decir, siguiendo de nuevo el ejemplo de las figuras, en el nivel intermedio la coordenada (x_2, y_2) se codifican como un valor diferencial relativo a la coorde-

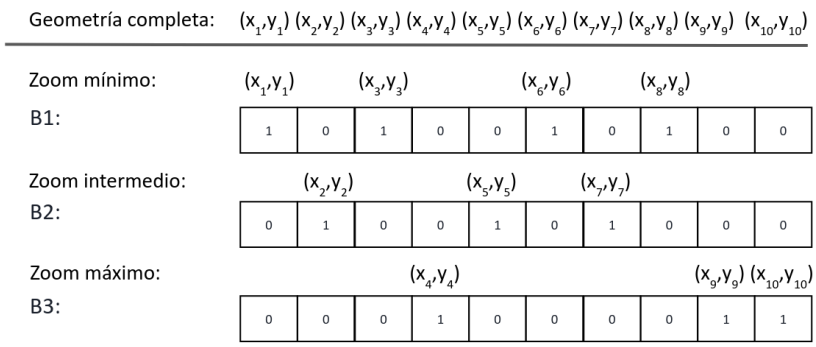

Figura 2. Ejemplo de la estructura correspondiendo a las geometrías de la Figura 1.

nada (x_1, y_1) en el nivel mínimo de zoom. Asimismo, (x_5, y_5) se codificarían en relación a los valores de (x_3, y_3) mientras que (x_7, y_7) sería relativo a (x_6, y_6). Por otro lado, (x_4, y_4) se codificaría como valores diferenciales respecto al valor absoluto de (x_2, y_2), el cual sería estaría precalculado respecto a (x_1, y_1). Lo mismo aplica al resto de puntos en el nivel.

3. Conclusiones y trabajo futuro

Nuestra propuesta puede almacenar las geometrías de un objeto geográfico divididas en varios niveles de zoom, y es asimismo posible recuperar la representación correspondiente a cualquiera de dichos niveles. En este artículo no se ha ahondado en la representación a nivel de implementación, pero es un aspecto que tiene un gran impacto sobre la eficiencia, tanto a nivel de almacenamiento como de tiempo de recuperación en la estructura. Igualmente, será necesario plantear el diseño de algoritmos de consulta adaptados a la estructura final que se desarrolle, para poder recuperar regiones específicas de las geometrías almacenadas de acuerdo con las interacciones de un usuario sobre el mapa dibujado en una aplicación cliente. También será necesario considerar las comunicaciones cliente-servidor para optimizar la transmisión de datos.

Referencias

1. L. Stoimenov and S. Djordjević-Kajan, "An architecture for open and scalable web-gis," in *Proc. of AGILE*, 2005.
2. S. Agrawal and R. Gupta, "Web gis and its architecture: a review," *Arab. J. Geosci.*, vol. 10, p. 518, 2017.
3. D. H. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica*, vol. 10, pp. 112–122, 1973.
4. M. Ajtai, R. C. Burns, R. Fagin, D. D. E. Long, and L. J. Stockmeyer, "Compactly encoding unstructured inputs with differential compression," *J. of the ACM*, vol. 49, no. 3, pp. 318–367, 2002.