Inicio
Sobre JCC
Programa
Eventos
Inscripción
Trabajos Aceptados
Charlas Invitadas
Organización
Planos de Ubicación
Nuestra Ciudad
Alojamiento
Contáctenos
English Version

Lista de Artículos aceptados para el Encuentro Chileno de Computación

1 **Implementación de un Almacén de Datos para DB2 usando instrucciones SQL: Una solución ROLAP.**
Angélica Urrutia Sepúlveda y Jessica Carolina Figueroa González.
Universidad Católica del Maule.

2 **Fábrica de Software: Hacia la Industrialización del Desarrollo de Software en la Minería.**
Broderick Crawford, Andrés Cabach, Mary Aranda.
Pontificia Universidad Católica de Valparaíso.

3 **Colonias de Hormigas en el problema de Timetabling: Aplicación al problema de horarios en la Facultad de Ingeniería.**
Colomba Machiavello, María Angélica Pinninghoff, Ricardo Contreras.
Universidad de Concepción.

4 **Definición de un Recuperador de Imágenes basado en Contenidos sobre Espacios Métricos.**
Eduardo Peña Jaramillo, Roberto Uribe Paredes.
Universidad de Magallanes.

5 **Índice espacio-temporal D*R-Tree: estudio experimental de su desempeño.**
María G. Dorzán, Edilma O. Gagliardi.
Universidad Nacional de San Luis.

6 **Diseño de una arquitectura de tuneles IP simultaneos multi-protocolo.**
Gustavo Chain Dumit, Rodrigo Ahumada Montenegro, José Miguel Rubio L.
Pontificia Universidad Catolica de Valparaíso.

7 **Reflexiones sobre el Software Libre; Ventajas, Desventajas y Amenazas.**
Jorge Benavides Escobillana, Héctor Beck Fernández.
Universidad de Tarapacá.

8 **Interacción en Ambientes Virtuales 3D utilizando el dispositivo "P5 DATA GLOVE".**
Hubert Hoffmann, Carol Chamblas.
Universidad Técnica Federico Santa María.

9 **JSP Programming Language and a Comparrison to PHP.**
Luis Marco Cáceres Álvarez.
Universidad de Tarapacá.

10 **Diseño e Implementación de un Laboratorio de Redes Remoto Para la Educación en Redes de Computadores.**
Marco A. Aravena Vivar, Andrés A. Ramos Magna.
Universidad de Valparaíso.

11 **Diseño e Implementación de un Simulador para el Proceso de Diseño de Software.**
Marta Barría, Víctor Bahamondes, Liliana Guzmán.
Universidad de Valparaíso.

Patrocinadores

Sociedad Chilena de Ciencia de la Computación

Gobierno de Chile Comisión Nacional de Investigación Científica y Tecnológica (CONICYT)

Auspiciadores

Compañía Minera Quebrada Blanca

COLLAHUASI

PEARSON Education

TERRADO SUITES HOTEL & MARINA

COMPUTRUX

YAHOO! RESEARCH

12 **SSSTree: búsqueda por similitud basada en clustering con centros espacialmente dispersos.**
Roberto Uribe-Paredes, Roberto Solar / **Universidad de Magallanes.**
Nieves R. Brisaboa, Oscar Pedreira, Diego Seco / **Universidad de A Coruña.**

13 **Sistema de Recomendación para apoyar la búsqueda de material educativo: un diseño basado en componentes y Patrones.**
Pedro G. Campos, Claudio Gutiérrez-Soto, María José Pérez Tobías, María José Romero Figueroa.
Universidad del Bío-Bío.

14 **Análisis Comparativo del Entrenamiento de Redes Neuronales Recurrentes para Sistemas Dinámicos.**
Rudy Velásquez E., Gonzalo Acuña L.
Universidad de Santiago de Chile.

15 **Contratos XML: un enfoque independiente para la detección y resolución de interacciones aspectuales.**
Sandra Casas / **Universidad Nacional de la Patagonia Austral.**
Claudia Marcos / **Research Institute. UNICEN.**

16 **Early Conflicts: Análisis y Resolución de Conflictos Tempranos.**
VeronicaVanoli / **Universidad Nacional de la Patagonia Austral.**
Claudia Marcos / **Research Institute. UNICEN.**

Arriba

RUNAS pub - restaurant

CASINO Iquique

AguaPura

VICENCIO PERFUMERIAS

Links de Interés

Alojamiento
Sernatur (Servicio Nacional de Turismo)
Iquique en Wikipedia
www.iquique.cl

Área Computación e Informática - Departamento de Ingeniería - Universidad Arturo Prat - Campus Playa Brava - Iquique - Chile
Teléfono: 56 (57) 394294 - (57) 394403 - (57) 394194 Fono-Fax: 56 (57) 394472 Email: jcc2007@unap.cl

Asunto: Artículo aceptado para el ECC'2007
De: "Marco Toranzo" <mtoranzo@unap.cl>
Fecha: Mon, 3 Sep 2007 15:51:40 -0400
Para: <ruribe@ona.fi.umag.cl>, <rsolarg@ona.fi.umag.cl>, <fbrisaboa@udc.es>, "Oscar Pedreira" <opedreira@udc.es>, <dsecog@udc.es>

Estimado autor(es) comunico a Usted que su artículo enviado al ECC´2007 fue ACEPTADO. Se adjunta archivo con los comentarios para mejorar el artículo. Se agradece su cooperación e interese por el ECC'2007.

Por favor, enviar el artículo con las recomendaciones hasta el dia 15 de septiembre.

Saludos cordiales,

Marco Toranzo

plantillaEvaluarArticulo_14.xls	**Content-Type:** application/vnd.ms-excel **Content-Encoding:** base64

SSSTree: búsqueda por similitud basada en clustering con centros espacialmente dispersos*

Roberto Uribe-Paredes[1,2], Roberto Solar[1], Nieves R. Brisaboa[3], Oscar Pedreira[3], Diego Seco[3]

[1] Dpto. Ingeniería en Computación, Universidad de Magallanes, Casilla 113-D, Punta Arenas, Chile

[2]Grupo de Bases de Datos (UART), Universidad Nacional de la Patagonia Austral, Rio Turbio, Santa Cruz, Argentina

[3]Laboratorio de Bases de Datos, Universidad de A Coruña, Campus de Elviña s/n, 15071 A Coruña, España

{ruribe,rsolar}@ona.fi.umag.cl, {brisaboa,opedreira,dseco}@udc.es

Resumen. *La búsqueda por similitud en espacios métricos es una operación fundamental en aplicaciones que trabajan con fuentes de datos no estructuradas. En este artículo presentamos un nuevo método de búsqueda por similitud basado en clustering, denominado SSSTree. Su principal característica es que los centros de cluster se seleccionan utilizando* Sparse Spatial Selection*, una técnica desarrollada originalmente para la selección de pivotes, capaz de adaptarse a la dimensionalidad intrínseca del espacio. Gracias a esto, el número de clusters en cada nodo depende de la complejidad del subespacio que tiene asociado. En este trabajo proporcionamos resultados preliminares con distintos espacios que demuestran que SSSTree obtiene mejores resultados que otras técnicas.*

1. Introducción

La búsqueda por similitud se ha convertido en una operación cada vez más importante para aplicaciones que trabajan con fuentes de datos no estructuradas. Por ejemplo, las bases de datos multimedia gestionan objetos sin ningún tipo de estructura, como imágenes, huellas dactilares o clips de audio. Recuperar la huella dactilar más semejante a una dada es un ejemplo de búsqueda por similitud. El problema de la recuperación de textos está presente en sistemas que van desde un simple editor hasta grandes buscadores. En este contexto puede ser interesante recuperar documentos similares a una consulta, o palabras muy similares a una dada para corregir errores de edición. Podemos encontrar más ejemplos de aplicación en áreas como la biología computacional (recuperación de secuencias de ADN) o el reconocimiento de patrones (donde un patrón puede clasificarse a partir de otros patrones similares ya clasificados) [Chávez et al., 2001, Zezula et al., 2006].

*Agradecimientos. Para N. Brisaboa, O. Pedreira y D. Seco: Trabajo parcialmente financiado por Ministerio de Educación e Ciencia (PGE y FEDER) ref. TIN2006-16071-C03-03 y (Programa FPU) ref. AP-2006-03214 (para Óscar Pedreira), y por Xunta de Galicia ref. PGIDIT05SIN10502PR y ref. 2006/4. Para R. Solar y R. Uribe: Fondecyt 1060776, Conicyt; programa de investigación PR-F1-002IC-06, Universidad de Magallanes, Chile y CYTEDGRID Proyecto 505PI0058, España.

El problema de la búsqueda por similitud puede formalizarse a través del concepto de espacio métrico. Un espacio métrico está formado por un universo de objetos y una función de distancia que determina la similitud entre dos objetos cualesquiera.

La implementación trivial de la búsqueda por similitud consiste en comparar la consulta con todos los objetos de la colección. El problema es que, en general, la evaluación de la función de distancia tiene un coste computacional elevado que hace que la búsqueda secuencial sea muy ineficiente. Esto ha dado lugar a una gran cantidad de trabajos en indexación y búsqueda en espacios métricos, que tratan de hacerla más eficiente y poder así aplicarla en grandes colecciones de datos. El objetivo de los algoritmos de indexación es reducir en todo lo posible el número de evaluaciones de la función de distancia. Esto se logra manteniendo en el índice información que, dada una consulta, permita descartar una gran cantidad de objetos sin compararlos directamente con la consulta.

Aunque este es el objetivo principal, hay otros aspectos que diferencian a los distintos métodos propuestos. Algunos métodos sólo permiten que la función de distancia tome valores discretos (como, por ejemplo, la distancia de edición). En cambio, otros fueron diseñados para trabajar con funciones de distancia continuas. Ésta es una cuestión importante, ya que restringe la aplicación del método a determinados dominios. Algunos métodos son estáticos, en el sentido de que la colección no puede crecer una vez creado el índice. En cambio, los métodos dinámicos soportan inserciones en la base de datos, o incluso crean el índice según se añaden objetos a la colección inicialmente vacía. Otro factor importante es la posibilidad de almacenar estas estructuras en memoria secundaria de forma eficiente, y el número de operaciones de E/S necesarias para acceder a ellas. Estas cuestiones no deben descuidarse debido a su importancia en la aplicabilidad y eficiencia del método.

En general, según [Chávez et al., 2001], hay dos clases de métodos de búsqueda, los basados en *clustering* y los basados en *pivotes*. Los *métodos de búsqueda basados en pivotes* seleccionan un conjunto de objetos de la colección como pivotes, y el índice se construye calculando la distancia entre éstos y cada elemento de la base de datos. En la búsqueda, esta información se utiliza para descartar objetos del resultado sin compararlos directamente con la consulta. Los *métodos basados en clustering* particionan el espacio métrico en un conjunto de regiones de equivalencia, cada una de ellas representada por un centro de *cluster*. En la búsqueda, se descartan regiones completas dependiendo de la distancia de su centro de *cluster* a la consulta.

Este artículo presenta una nueva estructura de indexación y búsqueda por similitud denominada SSSTree. Es un método basado en clustering, y su principal característica es que los centros de cluster se seleccionan aplicando *Sparse Spatial Selection* (SSS). SSS es una técnica desarrollada inicialmente para la selección de pivotes. SSS selecciona pivotes bien distribuidos en el espacio métrico, lo que da lugar a una mejora en el rendimiento de la búsqueda con respecto a una selección aleatoria. Además, presenta otras características importantes: es dinámica, ya que permite que la base de datos esté inicialmente vacía y crezca en el tiempo, y adaptativa, ya que no es necesario establecer de antemano el número de pivotes a utilizar y el propio algoritmo los selecciona según son necesarios para adaptarse a la complejidad del espacio. Nuestra hipótesis en este trabajo es que, al aplicar SSS para seleccionar los centros de los clusters, la partición del espacio será más eficiente y dará lugar a un resultado mejor en la operación de búsqueda. Los resultados

experimentales, aunque son preliminares, muestran que esta nueva aproximación obtiene un rendimiento mejor que otras propuestas anteriormente en dos colecciones de datos.

La Sección siguiente explica con más detalle conceptos previos sobre búsqueda por similitud. La Sección 3 presenta el problema de la selección de pivotes y centros en los algoritmos de indexación, y la técnica SSS, en que está basada la propuesta de este artículo. La Sección 4 describe SSSTree. En la Sección 5 se discuten los resultados experimentales preliminares obtenidos y 6 concluye el artículo con las conclusiones y trabajo futuro.

2. Trabajo relacionado

2.1. Búsqueda en espacios métricos

Como adelantabamos en la introducción, el problema de la búsqueda por similitud puede formalizarse mediante el concepto de espacio métrico. Un espacio métrico $(\mathbb{X}, d)$ está formado por un universo de objetos $\mathbb{X}$ y una función de distancia $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}^+$ que determina la similitud entre dos objetos cualesquiera. La colección o base de datos contiene un subconjunto finito de estos objetos $\mathbb{U} \subseteq \mathbb{X}$. Para que $(\mathbb{X}, d)$ sea un espacio métrico, la función de distancia debe satisfacer las siguientes propiedades: ser estrictamente positiva ($d(x, y) > 0$ y si $d(x, y) = 0$ entonces $x = y$), simétrica ($d(x, y) = d(y, x)$), y debe satisfaccer la desigualdad triangular

$$d(x, z) \leq d(x, y) + d(y, z)$$

La desigualdad triangular es la base de todos los algoritmos de indexación y búsqueda, ya que es la propiedad que permite descartar objetos del resultado utilizando distancias precalculadas, sin que sea necesario compararlos directamente con la consulta.

Una colección de palabras, con la distancia de edición (calculada como el número de caracteres a insertar, borrar o modificar para transformar una palabra en otra), es otro ejemplo de espacio métrico. Un espacio vectorial es un caso concreto de espacio métrico, en el que cada objeto está formado por k coordenadas. En este caso podemos utilizar, por ejemplo, cualquier función de distancia de la familia $L_s(x, y) = (\sum_{1 \leq i \leq k} |x_i - y_i|^s)^{\frac{1}{s}}$. Por ejemplo, L_1 es la distancia *Manhattan*, L_2 la distancia Euclídea, y $L_\infty = max_{1 \leq i \leq k} |x_i - y_i|$ la distancia máxima.

Hay varios tipos de consulta que se pueden realizar en un espacio métrico. La más común es la búsqueda por rango, que consiste en recuperar los objetos de la colección que están a menos de una distancia r dada de la consulta q, esto es $\{u \in \mathbb{U} / d(q, u) \leq r\}$. La búsqueda de los k-vecinos más cercanos consiste en recuperar los k objetos más semejantes a la consulta, esto es, el conjunto $A \subseteq \mathbb{U}$ tal que $|A| = k$ y $\forall\ u \in A,\ v \in \mathbb{U} - A,\ d(q, u) \leq d(q, v)$. La búsqueda por rango es la más general, y la de los k vecinos más cercanos puede implementarse en términos de esta [Chávez et al., 2001].

En un espacio vectorial, la dimensión del espacio es el número de componentes de cada vector. Aunque los espacios métricos generales no tienen una dimensionalidad explícita, se habla de su *dimensionalidad intrínseca* siguiendo la misma idea que en espacios vectoriales. Este interesante concepto permite distinguir espacios métricos de baja y alta dimensionalidad. La eficiencia de los algoritmos de búsqueda por similitud es peor en los espacios métricos con una dimensionalidad elevada. Por ejemplo, cualquier método de búsqueda se comportará peor en un espacio vectorial de dimensión 15 que en un espacio vectorial de dimensión 10. Esto se ha comprobado empíricamente en muchos trabajos realizados en espacios métricos. En [Chávez et al., 2001] se analiza con detalle este concepto y su influencia en la eficiencia de los métodos de búsqueda.

2.2. Algoritmos de búsqueda basados en pivotes

Los algoritmos basados en pivotes seleccionan en el espacio métrico un conjunto de puntos de referencia denominados pivotes. El índice creado mantiene en la estructura de datos que corresponda las distancias de estos pivotes a los objetos de la base de datos. Dada una consulta, estas distancias precalculadas que almacena el índice y las distancias de la consulta a los pivotes permiten descartar objetos del resultado sin compararlos directamente con la consulta. Por ejemplo, siendo p_i pivotes, q una consulta, r el radio de búsqueda, y $x \in \mathbb{U}$, si

$$|d(p_i, x) - d(p_i, q)| > r$$

podemos descartar x directamente, pues por la desigualdad triangular podemos deducir que $d(q, x) > r$.

Los métodos basados en pivotes más conocidos son *Burkhard-Keller-Tree* (BKT) [Burkhard and Keller, 1973], *Fixed-Queries Tree* (FQT) [Baeza-Yates et al., 1994], *Fixed-Height FQT* (FHQT) [Baeza-Yates, 1997], *Fixed-Queries Array* (FQA) [Chávez et al., 1999], *Vantage Point Tree* (VPT) [Yianilos, 1993] y sus variantes [Bozkaya and Ozsoyoglu, 1997] [Yianilos, 1999], *Approximating and Eliminating Search Algorithm* (AESA) [Vidal, 1986] y LAESA (*Linear AESA*) [Micó et al., 1994]. Recientemente se ha propuesto SSS [Brisaboa and Pedreira, 2007], una técnica de selección de pivotes que sirve de base a la técnica de selección de centros que se presenta en este artículo.

2.3. Algoritmos de búsqueda basados en clustering

Los algoritmos basados en clustering particionan el espacio métrico en varias regiones o clusters, cada uno de ellos representado por un centro de cluster. Así, el índice almacena la información sobre estos clusters de modo que, dada una consulta, puedan descartarse del resultado clusters completos comparando la consulta con los centros de cada cluster. En aquellos que no puedan descartarse, la consulta se compara secuencialmente con todos los objetos que pertenecen al cluster.

Existen dos criterios para delimitar las regiones en las estructuras basadas en clustering: hiperplanos, y radio cobertor. Los algoritmos que siguen el criterio de hiperplano dividen el espacio métrico en particiones de Voronoi. Dado un conjunto de centros de cluster $\{c_1, c_2, \ldots, c_n\} \subset \mathbb{X}$, el diagrama de Voronoi se define como la subdivisión del espacio en n áreas, de forma que $x \in Area(c_i)$ si y sólo si $d(x, c_i) < d(x, c_j)$, para $j \neq i$. En este tipo de algoritmos, dada una consulta (q, r), se descartan de la búsqueda aquellos clusters que no tienen ningún objeto en el resultado. Esta decisión se toma teniendo en cuenta el radio de búsqueda y la distancia de la consulta al hiperplano que separa a ese cluster del siguiente.

En el caso del radio cobertor, el espacio se divide en varias esferas que pueden intersectarse, y una consulta puede pertenecer a más de una esfera. El radio cobertor es la

distancia desde el centro del cluster hasta el objeto del cluster más alejado de él. Conociendo la distancia de la consulta al centro del cluster, el radio de búsqueda y el radio cobertor, podemos decidir si ese cluster no tiene objetos en el resultado. Los métodos basados en *clustering* más importantes son *Bisector Tree* (BST) [Kalantari and McDonald, 1983], *Generalized-Hyperplane Tree* (GHT) [Uhlmann, 1991] y *Geometric Near-neighbor Access Tree* (GNAT) [Brin, 1995].

BST [Kalantari and McDonald, 1983] particiona el espacio de forma recursiva almacenando la información sobre los clusters en un árbol binario. En la raiz se seleccionan dos centros de cluster y se realiza la partición, asignando los objetos del primer cluster al hijo izquierdo y los del segundo al derecho. Este proceso se repite en cada hijo para particionar el espacio de forma recursiva. Para cada cluster se almacena su radio cobertor y, durante la búsqueda, se procesa cada nodo del árbol si este radio determina que el resultado de la consulta intersecta con ese cluster. GHT [Uhlmann, 1991] particiona el espacio también de forma recursiva, pero durante la búsqueda no utiliza el radio cobertor para decidir qué clusters visitar. Para tomar esta decisión se basa en el hiperplano situado entre los dos centros de cluster almacenados en cada nodo. Estos dos algoritmos fueron los predecesores de GNAT.

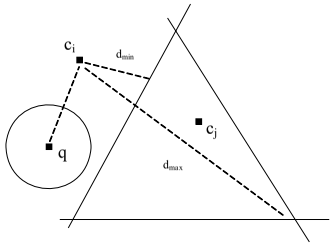


Figura 1. Uso de las distancias máxima y mínima para descartar clusters en GNAT

GNAT [Brin, 1995] también particiona el espacio de forma recursiva, pero en este caso se usan m centros de cluster en cada nodo interno del árbol, lo que da lugar a un árbol m-ario. Así, en cada nodo del árbol se divide el cluster que le corresponde en una partición de Voronoi. En cada nodo se almacena una tabla de m filas (una por cada centro de cluster) y m columnas (una por cada cluster). En la celda (i, j) se almacena $range(i, j)$, la distancia mínima y máxima del centro c_i a un objeto de $Cluster_j$. Dada una consulta q, se compara con un centro de cluster c_i. Así, se pueden descartar todos los clusters $Cluster_j$ tales que $d(q, c_i)$ no se encuentre entre las distancias mínima y máxima de $range(i, j)$. La figura 1 muestra de forma intuitiva el significado de estas distancias y cómo se pueden utilizar para decidir en qué clusters continuar la búsqueda y en cuales no. Las distancias máxima y mínima desde c_i a los objetos del cluster de c_j, y la distancia $d(q, c_i)$ y el radio de búsqueda r nos permiten saber que el cluster de c_j no tiene ningún objeto en el resultado (utilizando la desigualdad triangular).

El EGNAT [Uribe, 2005] pertenece al grupo de algoritmos basados en particiones compactas y es una optimización en memoria secundaria para el GNAT en términos de espacio, accesos a disco y evaluaciones de distancia. El EGNAT es un árbol que posee dos tipos de nodos, un nodo bucket y un nodo GNAT. Los nodos nacen como bolsas o buckets y cuya única información es sólo la distancia al padre, al estilo de las estructuras basadas en pivotes (un solo pivote). Este mecanismo es el que permite disminuir el espacio requerido en disco para almacenar la estructura y logra mantener un buen desempeño en términos de evaluaciones de distancia. Si un nodo se completa, éste evoluciona de un nodo bolsa a un nodo GNAT, reinsertando los objetos de la bolsa al nuevo nodo GNAT. Durante los procesos de busqueda, para descartar subarboles utiliza las tablas de rangos y para descartar objetos en las bolsas, la desigualdad triangular usando el padre como pivote.

En trabajos como [Chávez et al., 2001], [Zezula et al., 2006], y [Samet, 2006] puede encontrar muy buenas revisiones del problema de búsqueda por similitud en espacios métricos, que presentan con más detalle todo el marco teórico y los algoritmos que hemos mencionado.

3. Selección de pivotes y centros

Algo común en todos los algoritmos que hemos mencionado es que tanto los pivotes en unos como los centros de cluster en otros se seleccionan de forma aleatoria. Pero es evidente que el conjunto específico de puntos de referencia seleccionados tiene una fuerte influencia en la eficiencia de la operación de búsqueda. El número, su ubicación en el espacio métrico y su posición con respecto a los demás determinan la capacidad del índice para descartar objetos sin compararlos directamente con el objeto de consulta. Otro problema relacionado con estos puntos de referencia es cómo determinar el número óptimo de puntos a seleccionar. Por ejemplo, en el caso de los algoritmos basados en pivotes, podría parecer que cuantos más pivotes seleccionemos más objetos podremos descartar, pero también hay que comparar la consulta con los pivotes, por lo que hay que llegar a un compromiso entre el número de puntos de referencia y los objetos que podemos descartar con ellos. Todo esto hace que, si los puntos de referencia se seleccionan al azar, la calidad de los resultados obtenidos dependa del espacio y la colección concreta con la que se está trabajando, y del propio azar.

En algunos trabajos se han propuesto distintas heurísticas para la selección de pivotes. Por ejemplo, en [Micó et al., 1994] se propone seleccionar como pivotes los objetos que maximicen la suma de las distancias a los pivotes ya seleccionados. En [Yianilos, 1993] y [Brin, 1995] se siguen heurísticas para tratar de obtener pivotes lejanos entre si. En [Bustos et al., 2003] se trata en profundidad este problema y se demuestra empíricamente cómo este factor puede afectar al rendimiento de la operación de búsqueda. Además, [Bustos et al., 2003] propone un criterio para comparar la eficiencia de dos conjuntos de pivotes del mismo tamaño, y propone varias técnicas de selección de pivotes basadas en dicho criterio.

La primera de esas técnicas es *Selection*, que selecciona N conjuntos aleatorios de pivotes y se queda finalmente con aquel que maximice el valor del criterio de eficiencia. La siguiente es *Incremental*, una selección incremental en la que se añade al conjunto de pivotes aquel objeto de la colección que eleve más el criterio de eficiencia. La última es *Local Optimum*, en la que se parte de un conjunto aleatorio de pivotes y en cada iteración se sustituye el pivote que menos aporta al criterio de eficiencia por otro objeto. Aunque todas ellas son capaces de seleccionar pivotes que mejoran la eficiencia de la operación

de búsqueda, en todas hay que prefijar de antemano el número de pivotes. Así, hay que obtener el número óptimo de pivotes por prueba y error sobre una colección de datos significativa.

3.1. Sparse Spatial Selection (SSS)

Sparse Spatial Selection (SSS) [Brisaboa and Pedreira, 2007] es una técnica que selecciona de forma dinámica un conjunto de pivotes bien distribuidos en todo el espacio métrico. Se basa en la idea de que, si los pivotes están dispersos "espacialmente" en el espacio, serán capaces de descartar más objetos durante la búsqueda. Para lograr este objetivo, cuando un objeto se inserta en la base de datos, se selecciona como un nuevo pivote si está lo bastante lejos de los pivotes ya seleccionados. Se considera que está lo bastante lejos de otro pivote si está a una distancia mayor o igual que $M\alpha$ de este, siendo M la distancia máxima entre dos objetos cualesquiera y α un parámetro cuyos valores óptimos están en torno a $0{,}4$ (es decir, el objeto se selecciona como pivote si está a poco menos de la mitad de la distancia máxima de todos los demás pivotes). El parámetro α influye en el número de pivotes que se seleccionan. La figura 2 muestra con varias colecciones de datos que los valores óptimos de este parámetro siempre están entre $0{,}35$ y $0{,}40$, y que la eficiencia de la búsqueda es prácticamente la misma para todos los valores dentro de este intervalo.

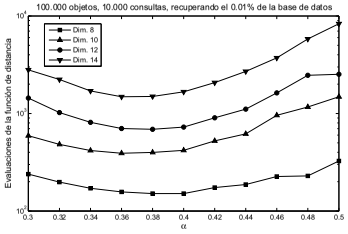


Figura 2. Eficiencia de la búsqueda en función del parámetro α

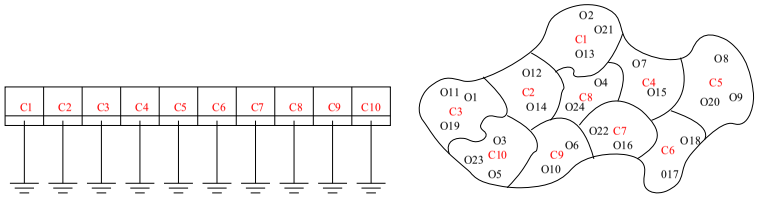
Los resultados que presenta [Brisaboa and Pedreira, 2007] demuestran que esta técnica es más eficiente que las anteriores en la mayoría de los casos. Además de ser eficiente, SSS posee otras características muy importantes. SSS es una técnica dinámica. Es decir, siguiendo el procedimiento de selección anterior, la base de datos puede estár inicialmente vacía, y los pivotes se seleccionarán cuando sea necesario según crece la base de datos. También es una técnica adaptativa. En SSS no es necesario establecer de antemano el número de pivotes a seleccionar. Cuando la base de datos crece, el algoritmo determina si la colección se ha vuelto lo bastante compleja como para seleccionar otro pivote. Así, SSS se adapta a la dimensionalidad intrínseca del espacio métrico [Brisaboa and Pedreira, 2007] (una medida de su complejidad).

En [Brisaboa and Pedreira, 2007] esta técnica de selección de pivotes se aplicó con distintas colecciones de prueba, dando lugar a una eficiencia un poco mejor que técnicas anteriores, a lo que suma las propiedades de ser una estrategia dinámica y adaptativa. Aunque originalmente se pensó como una técnica para la selección de pivotes, en este trabajo se ha utilizado para la selección de centros de cluster, basándose en la hipótesis de que, si los centros de cluster están bien distribuidos en el espacio métrico, la partición del espacio será mejor y la operación de búsqueda más eficiente.

4. SSSTree

SSSTree es un método de indexación basado en clustering con estructura de tipo árbol, en el que los centros de cluster de cada nodo interno del árbol se seleccionan aplicando *Sparse Spatial Selection* (SSS). La hipótesis en que nos basamos es que al utilizar estos centros de cluster, la partición del espacio será mejor y mejorará el rendimiento de la operación de búsqueda. En cada nodo se particiona el subespacio que le corresponde en varias particiones o clusters, y para cada uno de ellos se almacena el radio cobertor, utilizado durante la operación de búsqueda.

4.1. Construcción



(a) Centros seleccionados (b) Bucket auxiliar

Figura 3. Situación después de la elección de los centros del nodo raíz.

En el proceso de construcción se utiliza una estructura auxiliar denominada bolsa o *bucket*, que simplemente contiene un subconjunto de los objetos de la colección. Inicialmente, todos los objetos de la colección están en un *bucket*. El primer paso es estimar la distancia máxima M entre los objetos del espacio como se hacía en SSS para la selección de los pivotes. Hecho esto, se seleccionan los centros de los clusters aplicando SSS, es decir, el primer objeto del bucket es el primer centro. Para cada nuevo objeto del bucket, se selecciona como un nuevo centro si está a una distancia mayor que $M\alpha$ de los demás centros ya seleccionados, y en caso contrario se inserta en el bucket del cluster que le corresponda (el del centro al que esté más cercano). La figura 3 muestra un ejemplo de la construcción del árbol después de la elección de los centros del nodo raiz. La figura 4 muestra el árbol después de la inserción de cada objeto en el cluster que le corresponde.

El proceso se repite de forma recursiva en cada uno de los clusters. Ahora los elementos de cada cluster están también insertados en un bucket, y se procesan de la misma forma que antes. De nuevo hay que estimar la distancia máxima en cada bucket (pues la distancia máxima anterior será demasiado grande para este cluster). Además, la distancia máxima puede variar de cluster en cluster incluso en el mismo nivel. En el siguiente apartado explicamos diferentes formas de estimar la distancia máxima dentro de un cluster de forma eficiente. Una vez estimada esta distancia, se seleccionan los centros de los nuevos clusters, y se crean los nuevos subespacios. El proceso termina cuando un cluster tiene un número de elementos menor o igual que un umbral δ o, alternativamente,

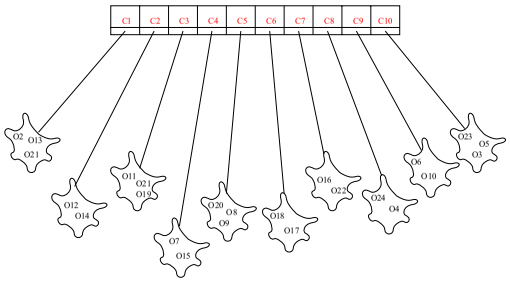


Figura 4. Nodo *SSSTree* después de la creación de los clusters.

cuando el radio cobertor del cluster es menor que un determinado umbral. En la búsqueda, el criterio para decidir en qué clusters continua la búsqueda es el de radio cobertor, que se almacena para cada cluster.

Al seleccionar los centros de cluster de esta forma, no todos los nodos tienen que tener el mismo número de hijos. Es decir, un cluster puede particionarse en un número de clusters distinto que otro debido a la distribución de los objetos en cada uno de ellos. Esta es una diferencia importante con estructuras anteriores como GNAT. La construcción del índice se adapta a la distribución de los objetos en el espacio métrico, y en cada nivel sólo se crearán aquellos clusters que se considere necesario. Esto se deriva del hecho de que SSS sea una técnica de selección capaz de adaptarse a la complejidad de cada espacio o subespacio.

4.2. Estimación del cálculo de M

Uno de los problemas del proceso de construcción anterior es la necesidad de calcular la distancia máxima M dentro de cada cluster, algo que sería muy ineficiente si lo tuviéramos que calcular con exactitud, debido a que habría que comparar todos los objetos del cluster entre si. Aunque la construcción del índice suele ser un proceso *off-line*, el valor de la distancia máxima puede estimarse adecuadamente para aligerar la construcción del índice.

M es la distancia máxima entre dos objetos cualesquiera del cluster, y el radio cobertor RC es la distancia del centro del cluster al objeto más alejado. Así, se cumple que $M \leq 2 \times RC$. Como puede observarse en la figura 5, al utilizar $2 \times RC$ como diámetro del cluster se logra cubrir los mismos objetos que al usar M como diámetro.

4.3. Búsqueda

Durante la operación de búsqueda se utiliza el radio cobertor de cada cluster para decidir en cada paso en qué clusters continuar buscando. El radio cobertor es la distancia entre el centro del cluster, c_i, y el objeto del cluster más alejado de él. Así, en cada paso se puede descartar el cluster de centro c_i si $d(q, c_i) - r > rc(c_i)$, donde $rc(c_i)$ es el radio cobertor de ese centro de cluster. Al llegar a las hojas del árbol que no pudieron ser descartadas, hay que comparar la consulta con todos los objetos asociados a esa hoja para obtener el resultado final de la búsqueda.

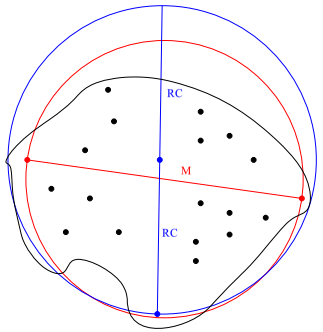


Figura 5. Optimización para el cálculo de M.

5. Resultados experimentales preliminares

En esta sección presentamos los resultados de pruebas preliminares que comparan el rendimiento de SSSTree en la operación de búsqueda con el de otros algoritmos basados en clustering. En las pruebas se han utilizado dos colecciones de datos:

- Espacios vectoriales sintéticos: Colección de $100{,}000$ vectores de dimensión 10, creados sintéticamente, con distribución gaussiana.
- Colección de palabras: En este caso se trata de un espacio métrico real, una colección de $86{,}061$ palabras extraídas del diccionario español.

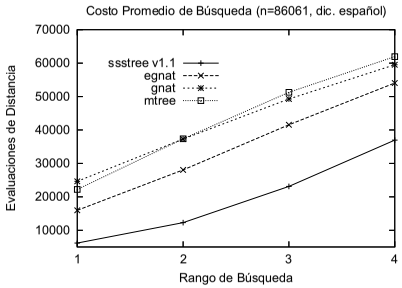


Figura 6. Evaluaciones de la función de distancia con M-Tree, GNAT, EGNAT y SSSTree, en una colección de palabras en español, para distintos rangos de búsqueda.

Los algoritmos con que hemos comparado SSSTree son M-Tree [Ciaccia et al., 1997], GNAT [Brin, 1995] y una variante de este, denominada EGNAT [Uribe, 2005]. La figura 6 muestra los resultados obtenidos en las pruebas realizadas con la colección de palabras en español. Como se puede ver en la figura, SSSTree logra un rendimiento mejor que todos los demás algoritmos con que se ha comparado en esta colección de prueba.

La figura 7 muestra los resultados obtenidos en las pruebas realizadas con la colección de vectores con distribución gaussiana. En este caso SSSTree presenta un rendimiento mejor que todos los algoritmos menos EGNAT.

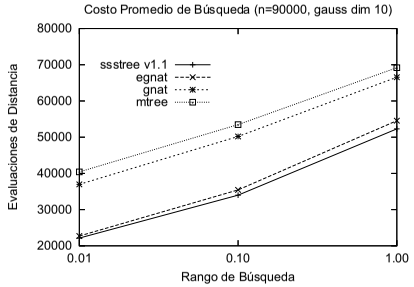


Figura 7. Evaluaciones de la función de distancia con M-Tree, GNAT, EGNAT y SSSTree, en una colección de vectores de dimensión 10 y distribución gaussiana.

Los resultados que presentamos en esta colección son sólo preliminares. En un futuro se harán comparaciones más exhaustivas con otros algoritmos teniendo en cuanto otros parámetros, como el tamaño de los clusters, y el número de particiones realizadas en cada nivel, y utilizando otras colecciones de prueba.

6. Conclusiones y trabajo futuro

Este artículo presenta un nuevo método de búsqueda en espacios métricos denominado SSSTree. Su característica más importante es que los centros de cluster se seleccionan aplicando *Sparse Spatial Selection*, una técnica desarrollada originalmente para la selección de pivotes. SSS es una estrategia adaptativa que selecciona puntos bien distribuidos en el espacio para lograr una eficiencia mayor. Estas dos propiedades se trasladan a SSSTree. Al seleccionar los centros de cluster de forma adaptativa, no tiene porque tener en cada nivel el mismo número de divisiones de cada cluster, si no las que se consideren necesarias en función de la complejidad de ese subespacio, una diferencia muy importante con todas las estructuras propuestas anteriormente.

El artículo también presenta resultados experimentales con vectores y una colección de palabras, que, aun siendo preliminares, muestran la eficiencia de SSSTree frente a otras técnicas. Estos resultados se deben al hecho de que los clusters están bien distribuidos en el espacio y de que tan sólo se crean aquellos clusters que se consideran adecuados en función de la complejidad del espacio métrico.

Nuestra línea de trabajo todavía mantiene abiertas algunas cuestiones que abordaremos en un futuro. En primer lugar, sería interesante evaluar el rendimiento de SSSTree con otros espacios métricos reales, como colecciones de documentos de texto o imágenes. Además, es importante estudiar en profundidad la condición de parada de la construcción del índice en función del radio cobertor del cluster en lugar del número de objetos que contiene. También sería interesante estudiar el comportamiento de SSSTree en colecciones de datos que contienen espacios métricos anidados [Brisaboa and Pedreira, 2007].

Referencias

[Baeza-Yates, 1997] Baeza-Yates, R. (1997). Searching: an algorithmic tour. *Encyclopedia of Computer Science and Technology*, 37:331–359.

[Baeza-Yates et al., 1994] Baeza-Yates, R., Cunto, W., Manber, U., and Wu, S. (1994). Proximity matching using fixed-queries trees. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, volume LNCS(807), pages 198–212.

[Bozkaya and Ozsoyoglu, 1997] Bozkaya, T. and Ozsoyoglu, M. (1997). Distance-based indexing for high-dimensional metric spaces. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1997)*, pages 357–368.

[Brin, 1995] Brin, S. (1995). Near neighbor search in large metric spaces. In *21st conference on Very Large Databases*.

[Brisaboa and Pedreira, 2007] Brisaboa, N. R. and Pedreira, O. (2007). Spatial selection of sparse pivots for similarity search in metric spaces. In *SOFSEM 2007: 33rd Conference on Current Trends in Theory and Practice of Computer Science*, LNCS (4362), pages 434–445.

[Burkhard and Keller, 1973] Burkhard, W. A. and Keller, R. M. (1973). Some approaches to best-match file searching. *Communications of the ACM*, 16:230–236.

[Bustos et al., 2003] Bustos, B., Navarro, G., and Chávez, E. (2003). Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366. Elsevier.

[Chávez et al., 1999] Chávez, E., Marroquín, J. L., and Navarro, G. (1999). Overcoming the curse of dimensionality. In *European Workshop on Content-based Multimedia Indexing (CBMI'99)*, pages 57–64.

[Chávez et al., 2001] Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. (2001). Searching in metric spaces. *ACM Computing Surveys*, 33:273–321.

[Ciaccia et al., 1997] Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435.

[Kalantari and McDonald, 1983] Kalantari, I. and McDonald, G. (1983). A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9:631–634.

[Micó et al., 1994] Micó, L., Oncina, J., and Vidal, R. E. (1994). A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, 15:9–17.

[Samet, 2006] Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann.

[Uhlmann, 1991] Uhlmann, J. K. (1991). Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179.

[Uribe, 2005] Uribe, R. (2005). Manipulación de estructuras métricas en memoria secundaria. Master's thesis, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile.

[Vidal, 1986] Vidal, E. (1986). An algorithm for finding nearest neighbors in (aproximately) constant average time. *Pattern Recognition Letters*, 4:145–157.

[Yianilos, 1993] Yianilos, P. (1993). Data structures and algorithms for nearest-neighbor search in general metric space. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321.

[Yianilos, 1999] Yianilos, P. (1999). Excluded middle vantage point forests for nearest neighbor search. In *Proceedings of the 6th DIMACS Implementation Challenge: Near neighbor searches (ALENEX 1999)*.

[Zezula et al., 2006] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity search. The metric space approach*, volume 32. Springer.