Informe Técnico ICC 2007-04-1

Métodos de Acceso para Bases de Datos Multimedia y sus Paralelizaciones

Fernando Artigas Fuentes, José Manuel Badía Contelles

Abril de 2007

Departamento de Ingeniería y Ciencia de Computadores

Correo electrónico: artigas@cerpamid.co.cu, badia@icc.uji.es

Universidad Jaime I
Campus de Riu Sec, s/n
12.071 - Castellón
España

Access Methods in Multimedia Databases and their Parallelization

Fernando Artigas Fuentes[1], José Manuel Badía Contelles[2]

Abstract:

Similarity queries are very important in data mining, and specially in text mining. The goal of this type of query is searching for all the objects in the database which are similar to a given object. Most similarity search techniques map the data objects into some feature space. Particularly in text mining this feature space has thousands of dimensions. Then, the similarity search correspond to a search of nearest-neighbor objects in the feature space.

On the other hand text databases are very large and so it is necessary to use indexing methods to organize the objects in the database in such a way that only a small portion of database must be explored to retrieve the target objects. This technical report reviews and compares the main tree indexing techniques, to determine their suitability to text mining in some stage of text processing. Most tree indexing methods perform badly in very-high-dimensional space, and so other techniques suitable for indexing in this type of space are also described. Our main conclusion is that only very few existing searching techniques (VA-File, IQ-Tree, Nene's method) can be applied to very high-dimensional spaces such as the spaces arising in text mining.

This report also reviews and compares the main parallel indexing techniques that appear in the literature.

Keywords:

access methods, multidimensional indexing, similarity indexing, high-dimensional indexing, paralellization, text mining, data mining.

[1]Centro de Estudios de Reconocimiento de Patrones y Minería de Datos, Universidad de Oriente, Santiago de Cuba, Cuba. E-mail: artigas@cerpamid.co.cu.

[2]Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, Castellón, España.

Métodos de Acceso para Bases de Datos Multimedia y sus Paralelizaciones

Fernando Artigas Fuentes[3] , José Manuel Badía Contelles[4]

Resumen:

Las consultas por semejanza son muy importantes en la minería de datos, y especialmente en la minería de textos. El objetivo de este tipo de consulta consiste en buscar en una base de datos todos los objetos que sean similares a uno dado. La mayoría de las técnicas de búsquedas por similitud usan una representación de los objetos reales dentro de un espacio de trabajo donde sus rasgos son dimensiones, por esto, las búsquedas por semejanza. se corresponden con la búsqueda de los objetos más cercano en el espacio de rasgos. En particular, para la minería de textos, este espacio de rasgos puede alcanzar miles de dimensiones.

Por otra parte, las bases de datos de textos suelen ser muy grandes, por lo que es necesario utilizar algún método de indexado para organizar los objetos en la base de datos de alguna forma que permita que sólo una pequeña porción de ésta sea explorada durante la recuperación los objetos en la solución a la consulta. Este informe revisa y compara las técnicas principales de indexado en forma de árboles, para determinar si son aplicables a la minería de textos en alguna de las fases de procesamiento. La mayoría de los métodos de indexado tienen malas prestaciones en espacios de muy alta dimensionalidad, por lo que también describimos otras técnicas apropiadas para el indexado en este tipo de espacio. Nuestra principal conclusión consiste en que pocas de las técnicas de búsqueda existentes (VA-file, IQ-tree, y el método de Nene) pueden ser aplicadas a espacios con muy alta dimensionalidad como los presentes en la minería de textos.

El informe también incluye una revisión y comparación de las principales técnicas paralelas de indexado que aparecen en la literatura.

Palabras clave:

métodos de acceso, indexado multidimensional, indexado por semejanza, indexado en alta dimensionalidad, paralelización, minería de textos, minería de datos.

[3]Centro de Estudios de Reconocimiento de Patrones y Minería de Datos, Universidad de Oriente, Santiago de Cuba, Cuba. E-mail: artigas@cerpamid.co.cu

[4]Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, Castellón, España.

1. Introducción

El objetivo principal de este informe consiste en realizar una comparación entre los métodos de indexado para el acceso a bases de datos de multimedia existentes en la literatura. Principalmente sobre su comportamiento con respecto a la dimensionalidad del espacio en que se representan los datos en las mismas. Tenemos interés especial en determinar los tipos de métodos son factibles de utilizar en la Minería de Textos, donde usualmente se trabaja con un número de dimensiones en el orden de las decenas de miles. Se presenta además un estudio de las versiones paralelas de los métodos más destacados.

Tradicionalmente las bases de datos se construyen alrededor de la idea de búsquedas exactas, los datos constituyen registros y cada uno de ellos contiene un campo clave completamente comparable. Como resultado de una consulta se devuelven aquellos registros cuyas claves coinciden exactamente con la que se busca. Una técnica muy utilizada en estos casos consiste en indexar los registros utilizando árboles como, por ejemplo, el B-tree [6, 31]. El uso de estas estructuras permite obtener los resultados de las búsquedas en tiempos menores a los obtenidos cuando se usan métodos de búsqueda secuencial.

El advenimiento de la era digital ha creado un interés creciente en la búsqueda de información en grandes repositorios que contienen datos mucho más complejos, como son los documentos textuales y datos multimedia. La búsqueda de objetos contenidos en estos repositorios no es exacta, sino que se necesita encontrar los objetos semejantes a uno dado. Para resolver este problema se aplican técnicas como son los árboles de indexado basados en semejanza.

Estos árboles requieren la definición de una métrica para comparar los objetos y se aprovecha la propiedad de la desigualdad triangular para encontrar los objetos más semejantes a uno dado, sin necesidad de calcular las semejanzas entre todos los pares de objetos.

En general, los distintos tipos de árboles de indexado se diferencian en la forma en que subdividen el espacio de representación de los objetos y el método de construcción del árbol.

Un tipo particular de árboles de indexado basados en semejanza son los árboles espaciales, que han sido ampliamente aplicados en Sistemas de Información Geográfica. En ellos se exige que los objetos estén representados por puntos en el espacio $\Re^n$ y utilizan esta propiedad para subdividir el espacio en regiones y facilitar la búsqueda. Ejemplos de árboles espaciales estudiados en la literatura son el Kd-tree [22], R-tree [27, 60] y X-tree [11]. Este tipo de árboles son muy sensibles al número de dimensiones del espacio de representación de los objetos y existen estudios [51] que demuestran que a partir de 10 dimensiones ya no son eficientes.

En la década de los 90 surge un nuevo tipo de árbol de indexado basado en semejanza que no exige una representación de los objetos en $\Re^n$ y que son algo menos sensibles al incremento del número de dimensiones. Ejemplos de estos árboles son el M-Tree [19, 17, 69], PM-Tree [63], entre otros. Un ejemplo de aplicación de este tipo de estructura lo tenemos en la recuperación de imágenes [12].

La determinación de los objetos semejantes a uno dado es un paso importante dentro de los algoritmos de agrupamiento tanto supervisados como no supervisados, por lo que el empleo de árboles de indexado es una buena opción para mejorar la eficiencia de estos algoritmos.

Los árboles de indexado basados en semejanza resultan de gran utilidad en problemas relacionados con el Procesamiento y Recuperación de Datos Multimedia [68], la Biometría [42] y en todos los problemas de Reconocimiento de Patrones que requieran de la búsqueda por semejanza. Pero por desgracia no son eficientes para el manejo de objetos representados en espacios de dimensionalidad muy alta, por lo que ha sido necesario el diseño de métodos de acceso específicos para este tipo de datos.

En muchas de estas aplicaciones se requiere procesar grandes volúmenes de datos con tiempos de respuesta acotados, por lo que un sólo ordenador no puede resolver el problema. Por lo general, un ordenador tampoco tiene la memoria necesaria para almacenar los datos que se necesitan procesar, por lo que hay que recurrir a datos en disco, lo que hace más lento el procesamiento.

Una solución usada en la actualidad consiste en resolver el problema explotando el poder de cómputo y la memoria disponible en las modernas arquitecturas paralelas. Así, la paralelización de los algoritmos de indexado posibilita la obtención de los objetos semejantes en un tiempo razonable cuando las colecciones son muy grandes, y por tanto, hacen más eficientes las técnicas que requieren este tipo de búsqueda.

En este informe se resume un conjunto importante de métodos de acceso para bases de datos en general, y se detallan los métodos de acceso fundamentales estudiados en la literatura para las bases de datos multimedia en particular. Se resumen además las principales técnicas de paralelización usadas en este campo.

El resto del informe se estructura del siguiente modo. En el apartado 2 se presentan los conceptos fundamentales relacionados con las consultas a bases de multimedia, con el objetivo de determinar los grados de semejanzas entre los objetos. En el apartado 3 se describen las características de los métodos de acceso y las principales aproximaciones. En el apartado 4 se describen las técnicas principales de paralelización de los métodos de acceso que han sido estudiadas en la literatura. Finalmente se presentan las conclusiones de este estudio, acerca de la factibilidad del uso de los métodos de acceso en varios procesos de la Minería de Textos.

2. Conceptos fundamentales

En este apartado se presentan los conceptos básicos relacionados con las consultas a bases de datos multimedia con el objetivo de determinar las semejanzas entre objetos presentes en estas y los objetos de consulta.

2.1. Conceptos preliminares

Objetos multimedia Se conoce como multimedia la combinación de texto, arte gráfico, sonido, animación y vídeo que es presentada por medio de un ordenador u otros medios electrónicos.

Los textos pueden aparecer con o sin formato, de manera lineal o como hipertextos. Los gráficos pueden representar esquemas, planos, dibujos lineales, etc. Las imágenes (documentos formados por pixeles) pueden tener como origen el escaneado, la fotografía digital, etc.

Las animaciones y vídeos presentan un número de imágenes por segundo que crean en el observador la sensación de movimiento. El sonido puede ser habla, música o de otro tipo.

La Multimedia está presente en campos como la Minería de Textos, la Biometría y en todos los problemas de Reconocimiento de Patrones.

Los objetos multimedia, llamémosles θ, para ser procesados por un ordenador, teniendo en cuenta la información que contienen, son transformados para ser representados en un espacio de datos determinado. La transformación más común consiste en convertirlos en puntos de un espacio vectorial multidimensional y definir la semejanza entre dos objetos con respecto a sus distancias en ese espacio vectorial.

Espacio de datos Es un espacio continuo de k dimensiones, $S_k \subset \Re^k$. Para este espacio se define una función Φ que hace de los objetos $\theta_i \subset \theta$, una abstracción, por lo que $\Phi(\theta_i) \rightarrow o_i, o_i \subset S_k$. La función Φ debe mantener, en S_k, entre los o_i las relaciones de semejanzas existentes entre los $\theta_i \subset \theta$.

Los o_i son los vectores que en una base de datos representan los objetos reales θ_i.

Base de Datos Una base de datos (BD) es una colección β de puntos k-dimensionales, x, tal que $x \in S_k$, por lo que $\beta = \Phi(\theta)$.

Teniendo en cuenta los conceptos anteriores, el problema de acceder y comparar los objetos reales, se transforma generalmente en el problema de acceder y comparar los vectores que representan sus características y que pueden estar contenidos o no en una BD.

2.2. Determinación de la semejanza

Para comparar dos objetos multimedia se necesita de una función de semejanza, que exprese de manera numérica cuan cercanos (o parecidos) son estos entre sí. Es posible utilizar diferentes medidas de similitud, pero hay que considerar que para diferentes medidas se obtendrán resultados diferentes.

Espacios métricos Una métrica es un tipo de función distancia utilizada para calcular la semejanza entre dos objetos. Definiendo esta función como ε tendremos que $\varepsilon : S_k \times S_k \rightarrow \Re^+$ y debe cumplir con las siguientes características:

a) Positividad: $\forall o_i, o_j \in S_k, \varepsilon(o_i, o_j) \geq 0$

b) Simetría: $\forall o_i, o_j \in S_k, \varepsilon(o_i, o_j) = \varepsilon(o_j, o_i)$

c) Reflexibidad: $\forall o_i, \in S_k, \varepsilon(o_i, o_i) = 0$

En la mayoría de los casos deberá cumplir además con

d) Positividad estricta: $\forall o_i, o_j \in S_k, o_i \neq o_j, \varepsilon(o_i, o_j) > 0$

Estas propiedades sólo aseguran una definición consistente. Para ser considerada una función distancia, ε deberá cumplir con la propiedad siguiente:

e) Desigualdad triangular: $\forall o_i, o_j o_k \in S_k, \varepsilon(o_i, o_j) \leq \varepsilon(o_i, o_z) + \varepsilon(o_z, o_j)$

El par (S_k, ε) es llamado *espacio métrico.*
Ejemplos de funciones métricas son las distancias Minkowiski definidas por:

$$\varepsilon(o_i, o_j) = L_p(o_i, o_j) = \sqrt[p]{(\sum_{k=1..n} \mid o_{i_k} - o_{j_k} \mid^n)}$$

En la Minería de Textos, para el cálculo de la semejanza entre documentos se pueden usar otras funciones, por ejemplo, la función coseno :

$$cos = \varepsilon(o_i, o_j) = \frac{\sum_{k=1..n} (o_{i_k} . o_{j_k})}{\sqrt{\sum_{k=1..n} o_{i_k}^2} . \sqrt{\sum_{k=1..n} o_{j_k}^2}}$$

donde n es el número de dimensiones (o términos) de los vectores.

La similitud máxima (igual a uno) se obtiene cuando todos los componentes de los vectores son iguales; y si no hay coincidencia en ningún elemento entonces se obtiene la mínima (igual a cero).

Debido a la probable alta dimensionalidad de los vectores de textos (cada palabra distinta en una colección de documentos es una dimensión), antes de procesar sus vectores, se procede a un proceso de filtrado conocido como reducción de la dimensionalidad, en el que se eliminan los términos superfluos y las palabras se agrupan teniendo en cuenta sus raíces.

Para calcular las distancias entre los objetos se puede usar:

$$\varepsilon(o_i, o_j) = \sqrt{1 - cos}.$$

debido a que cumple que los documentos más semejantes según coseno son los más cercanos según esa distancia. Es fácil demostrar que esta función es una métrica.

2.3. Consultas

Las BD se consultan para determinar cuál es el conjunto de sus objetos que pertenece a un hipervolumen $V \subset S_k$. V puede ser un punto, una hiperesfera, un hiperrectángulo, o cualquier otra figura geométrica.

En general, se distinguen dos tipos de consultas: las exactas y las aproximadas.

Consultas exactas A este caso pertenecen la consulta puntual, por un rango, o por vecindad.

La consulta puntual debe devolver, si existen, los objetos presentes en la BD cuyas distancias al objeto de consulta sean igual a cero, o sea:

Dado $q \in S_k, \forall o \in S_k$ si y sólo si $\varepsilon(q, o) = 0$

La consulta por rango determina cuáles son los objetos que se encuentran dentro de un hipervolumen definido por un objeto y una distancia de este. En este caso, dado $q \in S_k, r \in \Re^+, \forall o \in S_k$ si y sólo si $\varepsilon(q, o) \leq r$.

La consulta por vecindad determina cuales son los objetos presentes en la BD que están más cercanos al objeto consultado. Por lo general se determina un conjunto con k elementos, es decir, los k-vecinos más cercanos.

Las consultas exactas son útiles cuando se trabaja sobre espacios de trabajo discretos. En el caso de los espacios de datos multimedia, donde los objetos pertenecientes a las consultas, o los contenidos en las BD, pueden estar afectados por distintos tipos de ruidos, las consultas aproximadas son más eficientes.

Consultas aproximadas Las consultas aproximadas recuperan los objetos que están contenidos en el hipervolumen V con una probabilidad dada o un nivel de error limitado. En este caso las consultas de tipo puntual, por rango o por vecindad son realizadas usando una aproximación de la medida de semejanza.

2.3.1. Consultas por semejanza

La selección de objetos en una BD por semejanza es una generalización de las consultas de los k-vecinos más cercanos con un rango de búsqueda esférico [19]. Los parámetros involucrados en una consulta por semejanza son los siguientes:

- Un vector de consulta $q \in S_k$. El resultado de la consulta es ordenado en orden decreciente de distancia a q, donde la distancia usada es usualmente la euclidiana o la euclidiana ponderada.
- Un valor entero positivo k que especifica el número máximo vecinos más cercanos que debe ser devuelto en el resultado de la consulta.
- Un valor real positivo T que especifica la máxima distancia entre el vector de la consulta y cualquier vector en el resultado de la consulta.
- Un valor real no negativo ϵ que especifica el error máximo permitido en las aproximaciones. El error de aproximación viene dado por: $\frac{D}{\hat{D^*}} \leq 1 + \epsilon$, donde D es igual a T, si aparecen menos de k vecinos como respuesta a la consulta aproximada, o es la distancia desde q hasta el vecino número k, cuando este existe. $\hat{D^*}$ es la distancia desde q hasta su vector más cercano no incluido en la respuesta a la consulta aproximada. Este vector no incluido es aquel que no aparece en la consulta aproximada, pero sí aparece en la consulta exacta. Cuando $\epsilon = 0$, la consulta es exacta, ya que $\hat{D^*} = D$.

3. Métodos de acceso

En este apartado se presentan los conceptos relacionados con los métodos de acceso a los objetos en una BD, sus propiedades y clasificación; así como un resumen de las principales aproximaciones estudiadas en la literatura, con una comparación de sus prestaciones cuando trabajan con objetos representados en espacios de alta dimensionalidad.

3.1. Conceptos

Para cumplir con el propósito de realizar consultas en una BD, un método de acceso está formado por dos componentes principales: la estructura de indexado y la estrategia de búsqueda. Como convención, en el resto del informe, para referirnos a un método de acceso lo haremos usando el nombre de su estructura de indexado.

Estructura de indexado: Determina la organización de los datos que representan los objetos dentro de la BD. Por lo general esta almacena los datos en memoria externa. Para optimizar el acceso a memoria externa, por ejemplo un disco duro, cuando se lee o se escribe de esta, la unidad de datos utilizada no es el registro (que contiene la información de un sólo objeto), sino la página, que contiene un conjunto de registros y por lo tanto la información de varios objetos. Una vez que la página está cargada en memoria el acceso a los registros independientes es más rápido.

La lectura de información desde memoria externa incrementa el coste de los métodos de acceso. Tres aspectos fundamentales que caracterizan a una estructura de indexado, y que permiten minimizar este coste, son la capacidad de página, la capacidad de página efectiva y la selectividad. La primera es el número de objetos que puede almacenar una página; la segunda, es el número promedio real de objetos que son almacenados en las páginas; y la selectividad, es el porcentaje promedio de páginas del fichero que no son visitadas para responder a una consulta, es decir, la capacidad que tiene la estructura de evitar acceder a aquellos objetos que no pertenecen a la respuesta de la consulta.

La selectividad no sólo depende de la estructura, sino que está estrechamente relacionada con la estrategia de búsqueda y caracteriza la eficiencia de una estructura de indexado.

Estrategias de búsqueda: Determina el algoritmo con la que se recorre la estructura de indexado para optimizar el acceso a los objetos, con el objetivo de maximizar la selectividad y la precisión en los resultados a las consultas. Sobre cada tipo de estructura de indexado cada estrategia de búsqueda ofrece resultados diferentes. Una estrategia ideal para un caso puede no serlo en otro.

Las principales estrategias de búsquedas usadas en los métodos de acceso son:

Secuencial: Utiliza la fuerza bruta. Recorre los objetos uno a uno y determina si pertenecen o no a la respuesta de la consulta. Se toma como referencia para evaluar la eficiencia de otros métodos. Tiene una complejidad computacional de O(n), donde n es el número de objetos.

Con Particionamiento: La BD es dividida en particiones jerarquizadas. Permite leer sólo fracciones de la BD evitando aquellas que no intersectan con el hipervolumen de consulta. Se conocen para esta dos tipos de algoritmos: el RKV [52], con complejidad computacional O(log(n)); y el HS [28], con complejidad O(n).

Con múltiples etapas: Se usa cuando resulta muy costoso calcular sistemáticamente la distancia entre los objetos. En un primer paso se usa, en vez de la distancia real hasta cada objeto, la máxima distancia entre el objeto de la consulta y sus k-vecinos más cercanos. En una segunda etapa se seleccionan aquellos objetos que realmente están más cercanos al objeto de la consulta. Una versión óptima de este algoritmo se propone en [59].

Omni: Generaliza el concepto de búsqueda mediante el uso de objetos pivotes. Se aplica el algoritmo HF [48] que selecciona de forma heurística aquellos datos que proporcionan una buena cobertura del espacio de datos.

Aproximada: Puede realizarse de dos formas diferentes: deteniendo el proceso de búsqueda antes de que este se complete o descomponiendo el espacio de datos en k subespacios, generalmente unidimensionales, y realizando k búsquedas independientes; finalmente los objetos presentes en los k conjuntos de soluciones resultantes son comparados entre sí para obtener la solución definitiva.

Búsquedas en el espacio de solución: Consiste en indexar el espacio de solución. Este indexado puede realizarse, por ejemplo, aplicando una partición de Voronoi a la base de datos como se propone en [34].

3.2. Propiedades deseables

En general los métodos de acceso se espera que cumplan con las siguientes propiedades:

- Alta capacidad de página efectiva: El número de nodos de la estructura con ocupación por debajo de 50 % debe ser lo más bajo posible, con respecto al total de los nodos en la BD.
- Alta selectividad: Debe ser capaz de descartar el mayor número posible de nodos que no solapen con la solución de una consulta. La selectividad está estrechamente relacionada con el nivel de solapamiento entre los espacios de búsqueda. Es deseable que este solapamiento sea mínimo o no exista.
- Comportamiento escalable con la dimensionalidad: Debe ser capaz de mantener la eficiencia de la búsqueda aun cuando los espacios de búsqueda sean de alta dimensionalidad. La mayoría de los algoritmos propuestos se tornan ineficientes cuando la dimensionalidad del espacio de búsqueda es alta.
- Bajo índice de entrada/salida: Si los objetos de la BD están almacenados en disco, es deseable que el número de accesos a este sea lo menor posible. Si es necesario acceder con frecuencia, entonces es preferible usar algún forma de organización de los datos o preprocesado de estos que permita realizar accesos secuenciales en vez de aleatorios, que son menos eficientes.

- Algoritmo de agrupamiento eficiente: Si el método realiza particionado de los datos, es deseable usar un algoritmo de agrupamiento que permita que los objetos se almacenen en aquellos nodos donde se encuentren los objetos más similares a este. Esto puede lograse en el momento de la construcción inicial de la estructura de indexado, en las BD estáticas, o durante los procesos de inserción y eliminación de los objetos, en las BD dinámicas.
- Selección eficiente del pivote: Si el método de acceso realiza particionamiento del espacio, es deseable que el algoritmo de selección del objeto y la dimensión pivote se haga de forma eficiente.
- Tratamiento eficiente del exceso (overflow) y la carencia (underflow) de objetos en los nodos. Muchas estructuras definen los límites superior e inferior óptimos de ocupación de los nodos y realizan divisiones o fusiones de los mismos según sea el caso.
- Facilidad de paralelización: Esta característica es muy deseable cuando se trabaja con BD que contienen una gran cantidad de registros y donde se requieran tiempos acotados de respuesta a las consultas.

3.3. Clasificación de los métodos de acceso

Existe una forma clásica de clasificar los métodos de acceso (figura 1). Teniendo en cuenta las dimensiones del espacio de trabajo, se dividen en:

- Métodos de acceso de una dimensión.
- Métodos de acceso multidimensionales.

y usualmente se distingue una tercera categoría especial:

- Métodos de acceso de alta dimensionalidad.

Esta última clasificación responde al hecho de que está demostrado que no todos los métodos de acceso para espacios de datos de múltiples dimensiones escalan correctamente.

En los apartados siguientes nos referiremos principalmente a los métodos de acceso con estructura de árbol.

3.3.1. Métodos de acceso de una dimensión

Son los métodos desarrollados para las bases de datos numéricas tradicionales. Se clasifican en tres tipos: los métodos basados en tablas de dispersión, en las Curvas de Peano y en los árboles.

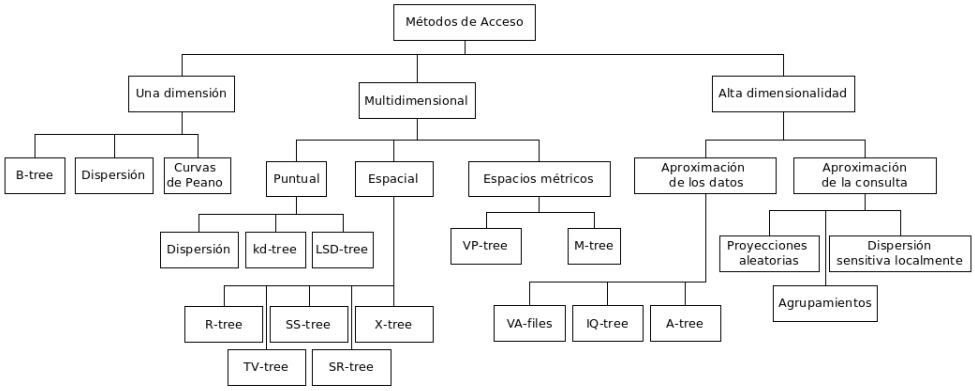

Figura 1: Clasificación general de los métodos de acceso

Tabla de dispersión: es una estructura que se utiliza para representar conjuntos grandes de valores que no se desean recorrer de forma secuencial y de los que se puede inducir una relación entre un conjunto pequeño de clases y el conjunto de valores. La tabla de dispersión garantiza operaciones con coste constante para el acceso y almacenamiento en el caso medio, aunque en el peor caso se comportan de forma lineal respecto al tamaño del conjunto. Se pueden usar distintos tipos de funciones para indexar los valores con el objetivo de garantizar una distribución homogénea de estos dentro de la tabla.

Curvas de Peano: son curvas que en su límite cubren todo el plano y tienen las siguientes propiedades:

- No pasan dos veces por el mismo punto.
- Son continuas y convergen uniformemente.
- La función que define la curva es inyectiva, son homeomorfas a un intervalo, sin embargo, el límite es de una dimensión superior.

Árboles: Son estructuras de indexado que contienen nodos conectados en una relación padre-hijo, el principal no tiene padre y es conocido como raíz; los nodos sin hijos se conocen como hojas. Los tiempos de acceso a sus elementos son de orden O(log n), donde n es el número de nodos en el árbol. Los principales tipos de árboles, para el espacio unidimensional, están divididos en dos grandes grupos:

- los balanceados por peso, por ejemplo, BB[α]-tree [32] y β-BBSTs-tree [16], y
- los balanceados por altura, entre las que se encuentran AVL-tree, y Scapegoat-tree [30].

También podemos citar los árboles de múltiples vías, como el B-tree [6, 31] y sus variantes.

3.3.2. Métodos de acceso multidimensionales

También conocidos como métodos espaciales, donde se destacan tres tipos:

- Los métodos de acceso puntuales.
- Los métodos de acceso espaciales.
- Los métodos de acceso de espacio métrico.

Métodos de acceso puntuales: Dividen de manera jerarquizada el espacio de búsqueda. A este grupo pertenece Kd-tree [22] y sus variantes, por ejemplo kdB-tree [33], 4D-tree, skd-tree [7], LSD-tree [1], GBD-tree [47], BV-tree, y más recientemente el Hybrid-tree [15].

Métodos de acceso espaciales: Utilizan métodos de particionamiento de datos. A este grupo pertenece R-tree [27, 60] y las variantes R+-tree [60] y R*-tree [44].

La estructura R-tree ha sido una de las que ha tenido mayor desarrollo y en [41] se presenta un estudio extenso de las nuevas variantes de R-tree, hasta el año 2005. En este estudio se agrupan las versiones en tres tipos:

- dinámicas
- estáticas
- espacio-temporales

En el primer grupo se incluye la estructura original y las variantes más simples. Recientemente se ha desarrollado el R-tree compacto [50] y el cR-tree [14].

En el segundo grupo se incluyen: R-tree empaquetado [52], Hilbert Packet R-tree [35], STR R-tree [38], Buffer R-tree [5] y R-tree con un número bajo de *stabbing* [20].

En el tercer y último grupo se encuentran: 3D R-tree [62], 2+3 R-tree [45], Historical R-tree [40, 45], R^{ST}-tree [55], MV3R-tree [61], R-tree parcialmente persistente [36], TB-tree [49] y finalmente el R-tree parametrizado en tiempo [56].

Otras estructuras inspiradas en R-tree son: Buddy-tree [58], TR*-tree [57, 37], TV-tree [39] y Cell-tree [26].

Métodos de acceso de espacio métrico: Como ya se había explicado, este tipo de método requiere de una función de distancia que cumpla con la desigualdad triangular. Pertenecen a este tipo la familia M-tree [18, 19, 17] y sus variantes M+-tree [66] y Pivoting M-tree [63]; y la estructura VP-tree [24].

3.3.3. Métodos de acceso híbridos.

Son estructuras que combinan las características de las anteriores para mejorar las prestaciones. Ejemplos de estructuras de esta clase son el VA-file [51], la técnica Pyramid [53], y más reciente, el IQ-tree [9].

3.3.4. Métodos de acceso para espacios de alta dimensionalidad

Se conoce como *maldición de la dimensionalidad* al problema de la pérdida de eficiencia de los métodos que particionan el espacio de datos, debido a que las particiones se incrementan de manera exponencial en relación con la dimensionalidad. La eficiencia disminuye al punto de que los métodos de acceso se comportan peor que si se usara la estrategia de búsqueda secuencial. Este problema ha sido demostrado de manera empírica y teórica por Weber et. al en [51].

Este problema afecta no sólo a la selectividad de la estructura, que tiende a cero y su complejidad supera a O(n), sino que también provoca que en determinados casos no se pueda comprobar que los k-vecinos más cercanos devueltos por una consulta sean significativamente más similares que el resto de los objetos presentes en la BD.

Dos de las soluciones propuestas para paliar el problema de la maldición de la dimensionalidad consisten en trabajar con una aproximación de los datos o con una aproximación de las consultas.

Aproximación de los datos: Consiste en reducir el número de dimensiones de los objetos mediante el uso de aproximaciones. De esta forma se reduce el número requerido de accesos a fichero, se incrementa la capacidad de página y mejora la selectividad del método.

Ejemplos de métodos de acceso que utilizan esta técnica son: VA-Files[64], IQ-Tree [10], A-tree [54] y la técnica Pyramid [53].

Aproximación de la consulta: Consiste en realizar consultas aproximadas siguiendo alguna de las estrategias ya comentadas: detención temprana de la búsqueda o proyección de los datos en varios sub-espacios de menor número de dimensiones.

Sobre proyección, recientemente se han presentado trabajos como el propuesto por Li Yang en [67] que conserva, en los datos proyectados las relaciones exactas de distancias entre los originales, pero no permite selección de parámetros por el usuario; la propuesta presentada en [21] por Deegalla, que hace un análisis de los componentes principales de los datos; y la realización de proyecciones aleatorias, como proponen Bingham et. al en [13].

Otras soluciones: Otra solución propuesta al problema de la dimensionalidad es presentada por Gionis en [25]. Consiste en aplicar una tabla de dispersión a los objetos de la BD para asegurar que la probabilidad de colisión sea mucho mayor entre los objetos cercanos que entre los objetos alejados entre sí. Este método ha demostrado ser más eficiente, por ejemplo, que el SR-tree para dimensiones cercanas a 50, en la búsqueda de soluciones aproximadas.

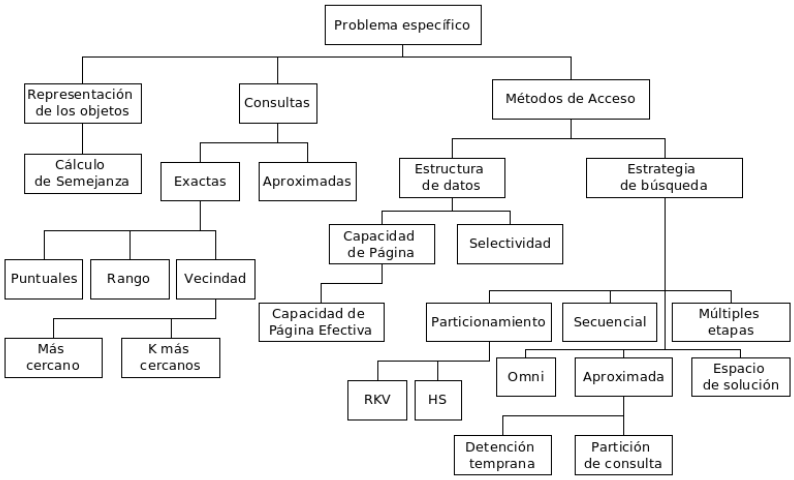

Figura 2: Aspectos a tener en cuenta para la evaluación de un método de acceso para su aplicación a problema en particular

3.4. Evaluación de los métodos de acceso

Como habíamos dicho en la introducción de este informe, nuestro objetivo principal consiste en evaluar las posibilidades que presentan los métodos de acceso para ser aplicados en la Minería de Texto.

En la figura 2 se resumen los aspectos a tener en cuenta para determinar si un método de acceso es apropiado para un tipo de problema en particular.

Primeramente es necesario conocer a fondo la naturaleza de los datos que se tratan en nuestro problema y los tipos de consultas que queremos hacerle a la BD. En la Minería de Textos, la característica fundamental consiste en la alta dimensionalidad de los datos, por lo que es necesario determinar la mejor representación de los mismos en la BD, de tal forma que la lectura de los vectores sea lo más eficiente posible y se reduzca el tiempo necesario para llevarlos a memoria principal; y la representación en esta última, para que el cálculo de la semejanza entre objetos se realice también en el menor tiempo posible. De los tipos de consultas, las más común en este tipo de problema consiste en determinar los vecinos más cercanos a un objeto dado.

Luego debe determinarse, de los métodos de acceso existentes, cuales poseen la estructura de datos y las estrategias de búsquedas más adecuadas para los rangos de dimensionalidad y distribución más frecuente en los datos que debemos procesar. Al final de este proceso se deben seleccionar aquellas que presenten una capacidad de página efectiva y una selectividad lo más altas posibles.

Finalmente debe estudiarse cuales son las estrategias de búsquedas que mejores resultados aportan según el tipo de consulta que debemos realizar.

La restricción más fuerte que la Minería de Textos impone a los métodos de acceso es la dimensionalidad, por lo que en el apartado siguiente haremos referencia a las pricipales aproximaciones estudiadas en la literatura y como se comportan con respecto a este aspecto.

3.5. Principales aproximaciones.

Una vez clasificados los distintos métodos de acceso, en este apartado mostraremos cómo se combinan las estructuras de datos y las estrategias de búsquedas en las principales aproximaciones que aparecen en la literatura. Se muestra además cómo influye en las prestaciones de estas la maldición de la dimensionalidad.

De [43] mostramos el cuadro 1 donde se agrupan los métodos de acceso por el rango de dimensionalidad en el que se comportan con eficiencia.

Método de acceso	Consulta	Estructura de datos	Estrategia de búsqueda
$d \leq 15$			
kd-tree	exacta, k-vecinos más cercanos	árbol de partición orientado a ISO	Partición del espacio
R*-tree	exacta, k-vecinos más cercanos	árbol MBR	Partición de datos
SS-tree	exacta, k-vecinos más cercanos	árbol MBS	Partición
TV-tree	exacta, k-vecinos más cercanos	árbol MBS con técnica de telescopio	Partición
VP-tree	exacta, k-vecinos más cercanos	árbol con puntos de anclaje	Omni
VP-tree	exacta, k-vecinos más cercanos	árbol con puntos de anclaje	Omni
M-tree	exacta, k-vecinos más cercanos	árbol con espacio métrico en forma de esferas	Partición y Omni
$d \leq 25$			
X-tree	exacta, k-vecinos más cercanos	Similar a R*-tree con nodos extendidos	Partición
$d \geq 25$			
VA-File	exacta, k-vecinos más cercanos	Aproximación de los datos en los ficheros	Múltiples pasos
IQ-tree	exacta, k-vecinos más cercanos	Similar a R*-tree con aproximación de los datos en los ficheros	Partición y múltiples pasos.
A-Tree	exacta, k-vecinos más cercanos	R*-tree con aproximación de datos y MBR.	Partición
Pyramid	exacta, por rango	Codificación Pyramid con B+-tree	Múltiples pasos

Cuadro 1: Métodos de acceso agrupados por el rango de dimensionalidad en el que se comportan con eficiencia

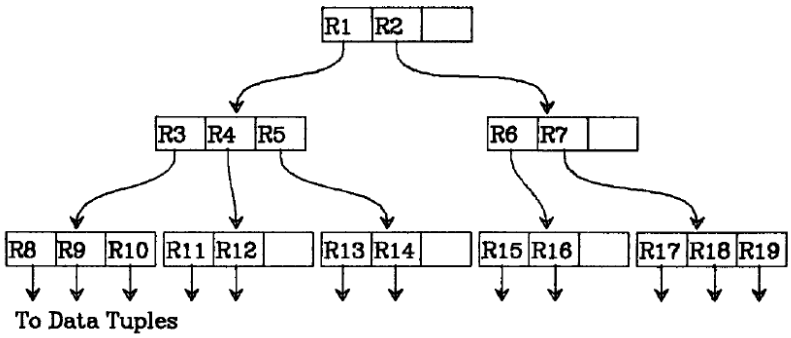

Figura 3: Organización de los objetos en un árbol R-tree.

En los siguientes apartados se describen las estructuras que podrían ofrecer mejores resultados en el campo de la Minería de Texto, como son el R-tree, el M-tree, VA-file e IQ-tree. Estos métodos son usados frecuentemente en la literatura como puntos de referencia para analizar el comportamiento de los métodos de acceso en cuanto a la dimensionalidad.

3.5.1. R-tree

El R-tree es una estructura dinámica de indexado para búsquedas espaciales. Organiza los datos en forma de árbol jerárquico de múltiples vías y los objetos son representados por hipercajas de mínimo cubrimiento. Estas hipercajas son cubiertas a su vez por otras hipercajas contenidas en niveles superiores del árbol, hasta que se llega al número de dos en la raíz de este (figura 3 [27]).

La inserción de un nuevo elemento se realiza en aquellas hipercajas que necesiten menor crecimiento para contenerlo. La búsqueda se realiza sobre las ramas del árbol que contienen hipercajas que solapan con la hipercaja del elemento buscado (figura 4 [27]).

En la figura 5 se describe el comportamiento del R-tree con respecto a la dimensionalidad del espacio de búsqueda. Como se aprecia, para dimensiones mayores de 10 en esta estructura es necesario visitar el 100 % de los nodos del árbol para dar respuesta a una consulta.

3.5.2. X-tree

En 1996 Stefan Berchtold presentó un nuevo método de indexado denominado X-tree [11]. Este método aportó una solución al problema del solapamiento de las hipercajas de cubrimiento presente en la familia R-tree. Para lograr esto propuso una nueva organización del árbol, en la que se utiliza un algoritmo de inserción que introduce el concepto de supernodos, a la vez que minimiza la necesidad de particionamiento y evita el solapamiento.

En el X-tree, los nodos de datos contienen hipercajas de cubrimiento mínimo (MBRs) y apuntadores a los datos reales de los objetos; y los nodos directorios contienen tanto MBRs cómo apuntadores a sub-MBRs (figura 6 [11]).

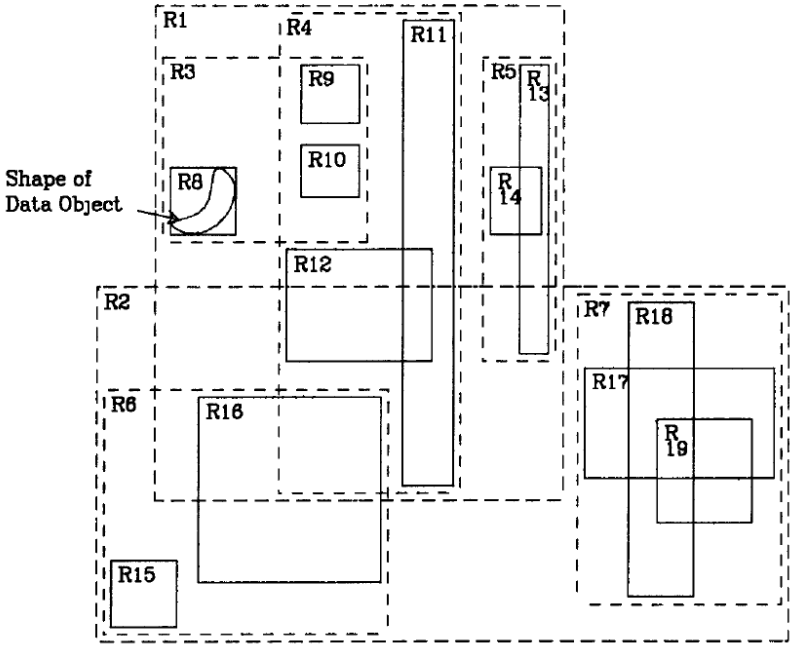

Figura 4: Hipercajas en un R-tree.

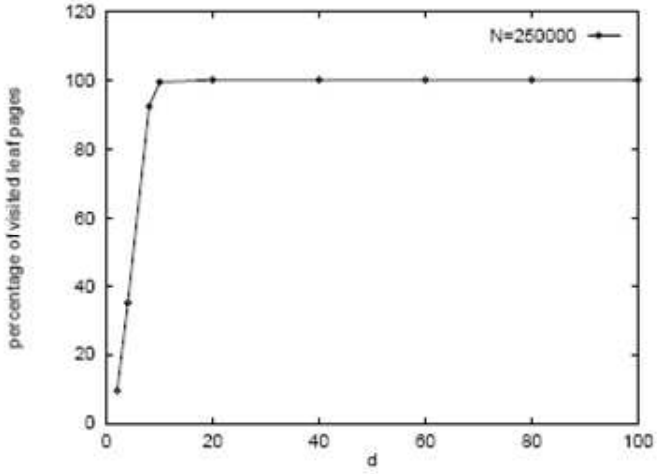

Figura 5: Comportamiento del R-tree con respecto a la dimensionalidad del espacio.

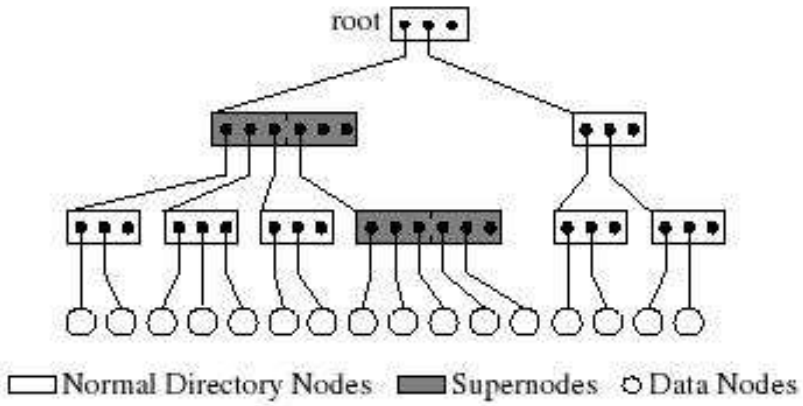

Figura 6: Estructura del X-tree

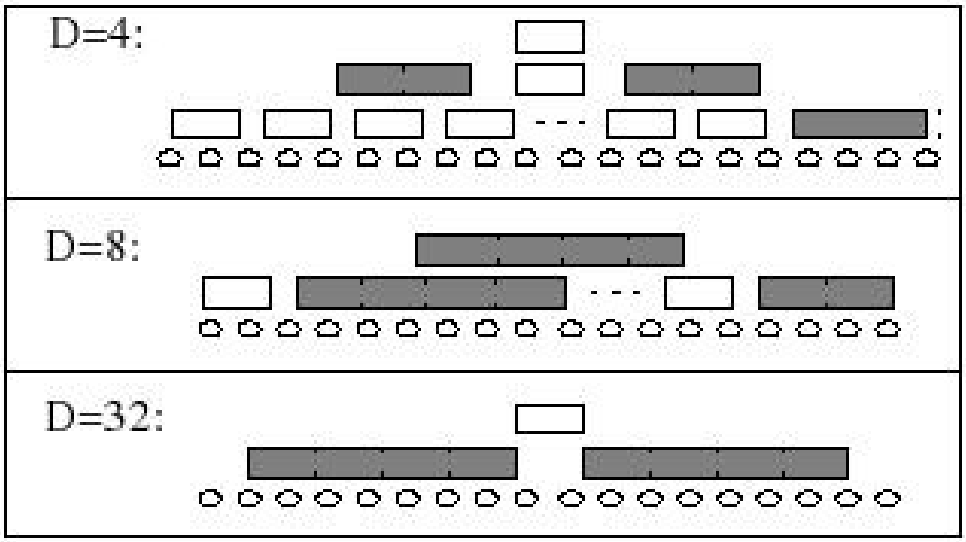

Figura 7: Configuración del X-tree para diferentes dimensiones de los datos

La estructura interna del X-tree varía de acuerdo con la dimensionalidad de los datos. En la figura 7 [11] se muestra como sería para datos de 4, 8 y 32 dimensiones.

Para demostrar la eficiencia de este método, se comparó con el TV-tree y el R*-tree usando datos sintetizados (figura 8 [11]). Sólo se muestran los resultados obtenidos para datos de hasta 32 dimensiones y, aunque este método representa una mejora apreciable con respecto al R*-tree, el número de páginas accedidas se incrementa con la dimensionalidad de los datos.

3.5.3. M-tree

M-tree es un método de acceso dinámico capaz de indexar espacios métricos, donde la función usada para calcular la distancia entre cualquier par de objetos satisface los postulados de la positividad, la simetría y la desigualdad triangular [18]. Para indexar los objetos, el

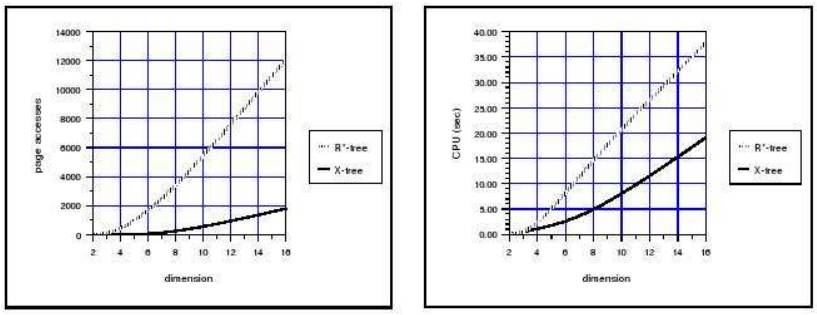

Figura 8: Comportamiento del X-tree con respecto al R*-tree, comparando el número de páginas accedidas y el uso del procesador con respecto a la dimensión de los datos.

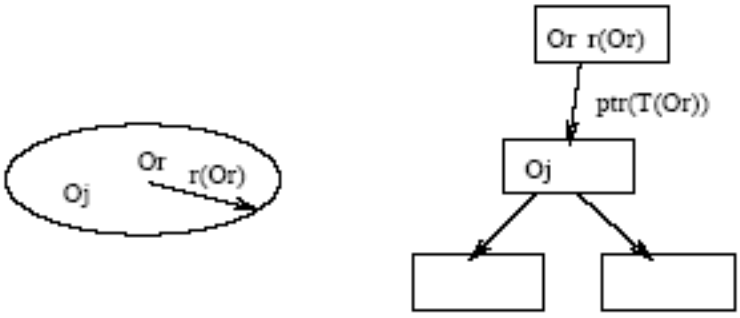

Figura 9: Un objeto de encaminamiento, O_r, tiene un radio de cobertura r(O_r), y hace referencia a un árbol de cobertura T(O_r).

usuario sólo tiene que suministrar las características que describen al objeto y la función para el cálculo de la distancia.

Los objetos pueden ser insertados en el M-tree uno a uno o usando una técnica de carga por grupos (bulkload) [17]. Al insertar, el usuario puede especificar la política que será aplicada cuando un nodo va a ser dividido en caso de que ocurra un desbordamiento. Si la carga se hace por grupos, entonces el usuario puede especificar tanto la utilización mínima de los nodos como la máxima.

Las búsquedas en el M-tree pueden ser de tres tipos: los k-vecinos más cercanos, los menores a un radio de búsqueda y accesos ordenados. El último caso puede verse como una búsqueda interactiva en la que el usuario va obteniendo uno a uno los próximos vecinos más cercanos al objeto de consulta.

El M-tree organiza los objetos en nodos de tamaño fijo (hasta M entradas) diferenciando dos tipos: los nodos de encaminamiento (en la raíz del árbol sólo pueden haber dos como máximo) y los nodos hojas, donde se almacenan los objetos. Cada nodo representa una cobertura del espacio métrico en forma de hiperesfera (figura 9 [18]).

En la figura 10 [18] se muestra para el M-tree el comportamiento de la selectividad con respecto al incremento de la dimensionalidad del espacio de representación de los objetos. Como se aprecia, a medida que aumenta el número de dimensiones, se incrementa también

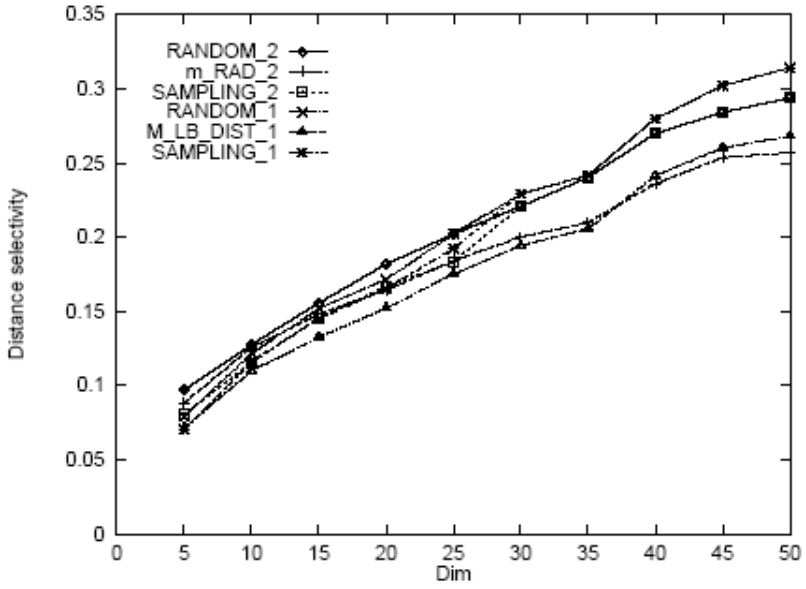

Figura 10: Selectividad del M-tree para obtener los primeros 10 vecinos más cercanos.

el número de nodos visitados (número de distancias calculadas) para dar respuesta a una consulta, por lo que disminuye la selectividad de la estructura. Es por eso que no se recomienda su uso en problemas mayores de 25 dimensiones.

Algunas variantes han sido propuestas para el M-tree, por ejemplo, el M+-tree[66], que mejora al M-tree usando un concepto llamado dimensión clave (*key dimension*), que efectivamente reduce el tiempo de respuesta a las consultas. Esta mejora consiste en separar los subespacios en dos, en los llamados nodos gemelos. En la figura 11 [66] se muestra la diferencia entre la división de los nodos en el M+-tree con respecto al M-tree.

Otra propuesta es el Pivoting M-tree (PM-tree) [63] que explota la idea de reducir el solapamiento entre las regiones reduciéndolas mediante el cambio de la hiperesfera por otra figura geométrica más compleja formada por hiperanillos (figura 12 [63]).

En ambos casos el objetivo consiste en reducir el solapamiento entre las regiones cuando se realiza una consulta, por lo que se incrementa la selectividad de la estructura.

3.5.4. VA-file

La mayoría de los autores que utilizan estructuras de indexado para trabajar sobre espacios multidimensionales consideran algunas decenas dimensiones como un número relativamente alto. Pero existen problemas en los que es necesario indexar objetos que pertenecen a espacios con más alta dimensionalidad. Por ejemplo, en la Minería de Textos, cada término (palabra) diferente en una colección de documentos es considerado como una dimensión, por lo que en colecciones extensas el número de dimensiones puede alcanzar las decenas de miles.

Para las estructuras descritas hasta el momento, a medida que aumenta el número de dimensiones, la selectividad va disminuyendo, hasta que se torna igual o peor que para la

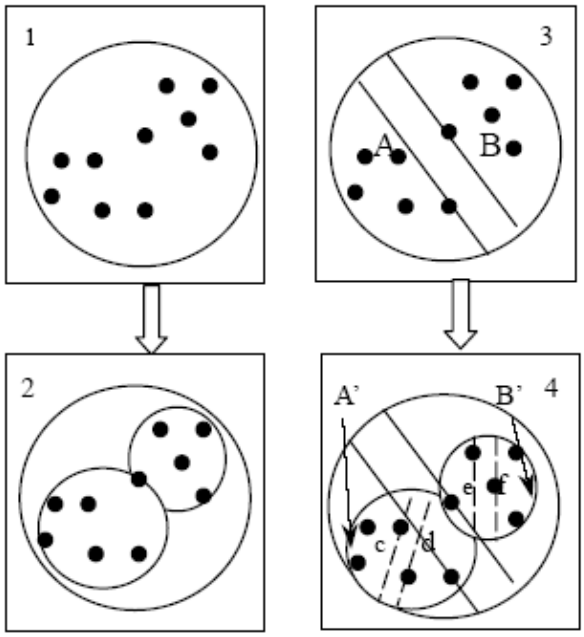

Figura 11: División de los nodos en M-tree (1 y 2) vs M+-tree (3 y 4).

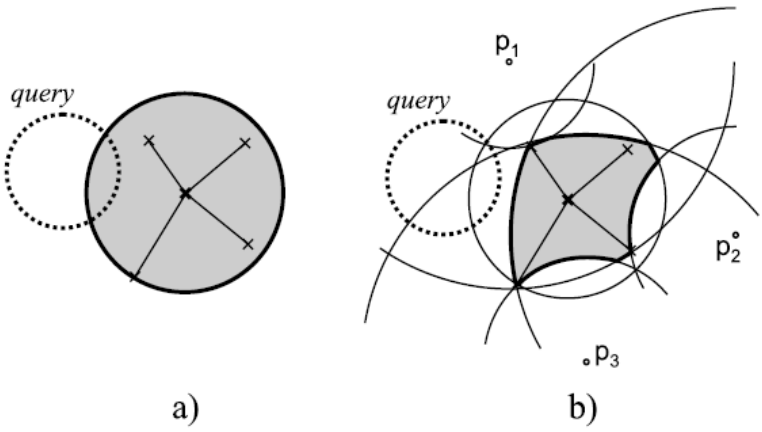

Figura 12: Regiones en el M-tree (a) vs regiones en el PM-tree (b)

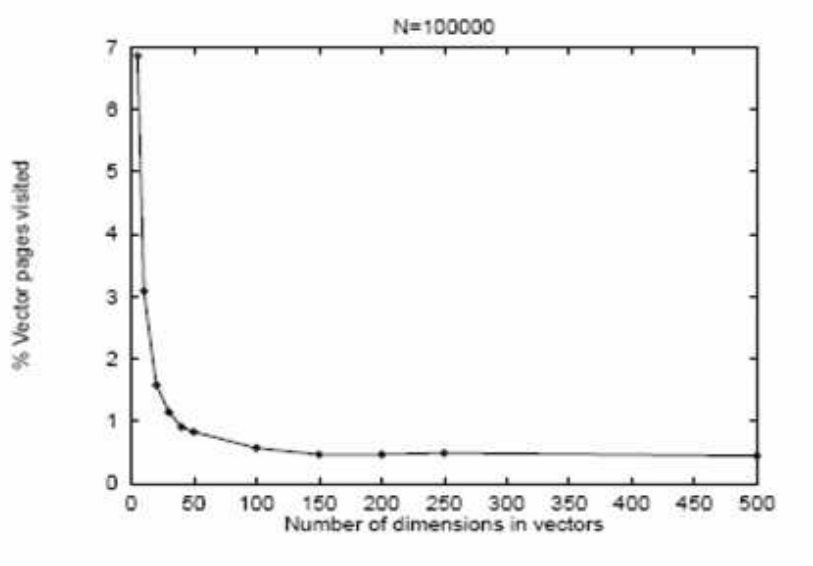

Figura 13: VA-file % de páginas visitadas vs. dimensionalidad

búsqueda secuencial.

Una solución al problema de la dimensionalidad fue la propuesta de Weber y Blott [64] de no seguir el tradicional método de particionar los datos, sino de aplicarles un proceso de filtrado.

El VA-file usa una aproximación geométrica compacta de cada vector. Por esta razón, algunos autores consideran que VA-file no es una estructura de indexado sino una técnica de compactación. Las aproximaciones son mantenidas en un fichero aparte al de los vectores originales. Ninguno de los dos ficheros es ordenado, pero cada objeto mantiene la misma posición en ambos.

Para realizar las búsquedas primero recorre las aproximaciones, proceso en que sólo un pequeño número de vectores es visitado, actuando como un filtro; luego realiza el cálculo de distancias sobre los pocos vectores originales resultantes de la primera etapa.

La ventaja principal de este método consiste en que su selectividad mejora a medida que crece el número de dimensiones (figura 13 [51]).

De este método se ha reportado en la literatura un estudio de paralelización que será tratado en el apartado 4.

3.5.5. IQ-tree

IQ-tree [10] propone la combinación de los métodos de indexado con los de compresión para la búsqueda de los vecinos más cercanos. Esta estructura posee tres niveles. El primer nivel es un directorio ordinario que contiene cajas de mínimo cubrimiento, el segundo nivel contiene representaciones comprimidas de los vectores y el tercer nivel contiene los datos reales (figura 14).

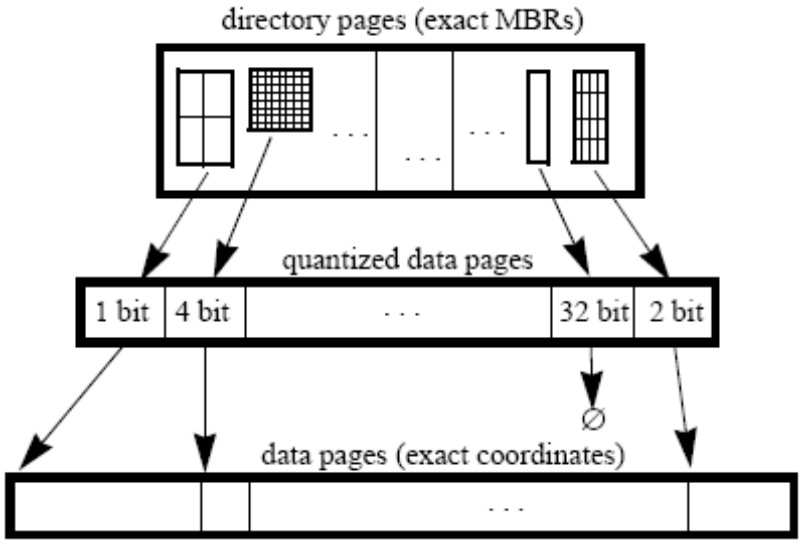

Figura 14: Estructura del IQ-tree

Para construir el árbol se sigue una técnica *top-down*, pero no es necesario continuar dividiendo hasta que se alcancen los datos en los nodos hojas, en vez de esto se selecciona un punto de parada apropiado. Los datos contenidos en cada región son codificados en una representación de bits, pero a diferencia que con el VA-file, para cada región se determina un esquema de representación independiente, de acuerdo con la densidad particular de puntos en la región.

Los resultados mostrados en [10] para el IQ-tree muestran mejores prestaciones que las obtenidas con VA-file y X-tree, para pruebas experimentales con datos de hasta 16 dimensiones, teniendo en cuenta el tiempo de obtención de las respuestas a las consultas (figura 15 [10]).

3.5.6. Otras propuestas

En el año 1975 Friedman propuso un algoritmo para obtener el vecino más cercano [23] basado en la aproximación de la consulta mediante proyecciones. Este algoritmo tiene dos etapas. En la primera, la de preprocesamiento, los objetos del conjunto inicial son ordenados de manera independiente por cada una de sus coordenadas. De esta forma cada una de las dimensiones se puede recorrer de manera individual como un vector unidimensional.

La segunda etapa es la de búsqueda, en la que a partir de un objeto nuevo, no contenido en el conjunto inicial, se obtienen sus vecinos más cercanos. Para esto, en cada vector unidimensional se determina un segmento, obtenido por la adición y substracción de un valor pequeño ε a cada una de las coordenadas del objeto buscado (figura 16 [23]). Luego se selecciona el segmento que contiene menos objetos. Finalmente, se realiza una búsqueda exhaustiva para determinar cuales de esos objetos son los más cercanos al de búsqueda.

Este algoritmo es claramente ineficiente para dimensiones altas ya que tiene una complejidad $O(nd\varepsilon)$. Por esto Nene y Naya propusieron [46] en 1997 una mejora para encontrar la

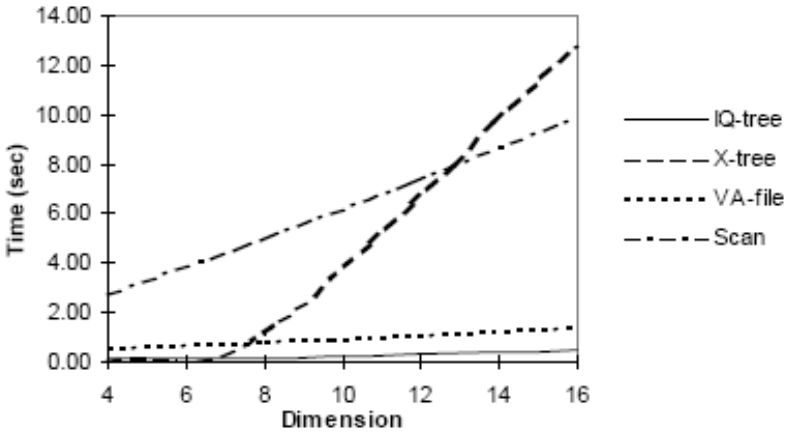

Figura 15: Comparación de IQ-tree, VA-file, X-tree y secuencial en cuanto a tiempo de obtención de respuesta a consultas.

lista de objetos candidatos a la búsqueda final. Esta mejora se basa en el uso de una estructura de datos preconstruida que presupone que el conjunto de objetos de entrenamiento es estático, y por lo tanto debe ser construida una sola vez.

En la propuesta sólo se realiza ordenamiento en la primera de las coordenadas y el conjunto inicial de candidatos está formado por los objetos encerrados dentro de los hiperplanos obtenidos para esa coordenada. Luego se sigue un proceso de descarte de los objetos usando el resto de las coordenadas. Las operaciones usadas en esta última etapa son comparaciones entre enteros y búsquedas en memoria.

Esta propuesta padece del problema de la selección correcta del valor ε, debido a que un valor muy pequeño puede tener como resultado un conjunto de solución vacío; y un valor muy grande impacta negativamente en las prestaciones de la misma. En el trabajo los autores proponen mecanismos para determinar el valor óptimo de ε.

Como resultado obtienen un algoritmo cuya complejidad es independiente de la dimensionalidad del problema, pero dependiente del valor de ε y del tipo de distribución de los objetos.

En espacios de alta dimensionalidad, este algoritmo tiene mejores prestaciones cuando la distribución de los datos es normal (figura 17 [46]), y empeora cuando la distribución de los datos es uniforme (figura 18 [46]).

3.5.7. Conclusiones

En resumen, de los métodos de acceso presentados, sólo son útiles para el trabajo en alta dimensionalidad el VA-File y el IQ-tree, que como se ha mostrado, mejoran la selectividad a medida que se incrementan las dimensiones del espacio de representación de los objetos; y el método de Nene, con un coste independiente de la dimensionalidad. El resto de los métodos empeoran sus prestaciones a partir de un cierto (y relativamente bajo) número de dimensiones por lo que sólo sería interesante aplicarlos, por ejemplo, en la representación

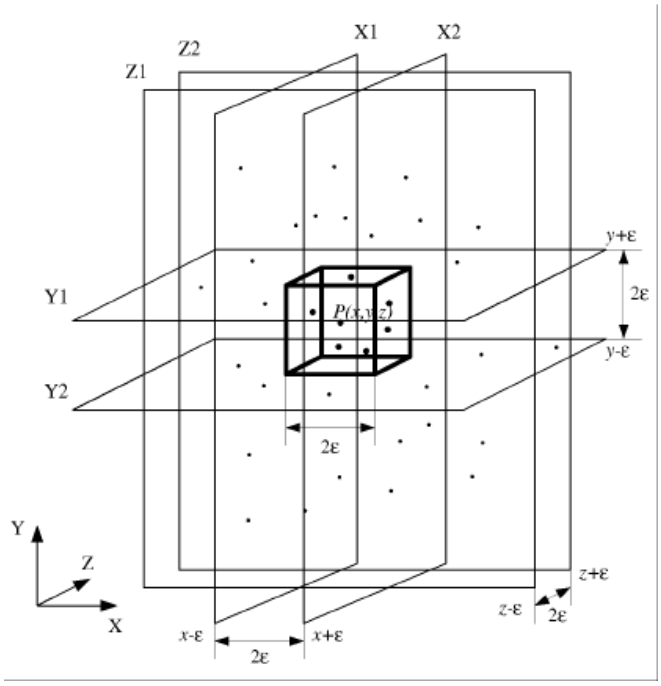

Figura 16: Hipercubo creado por el algoritmo de Friedman, determinado por la adición y sustracción del valor ε a cada coordenada del objeto buscado.

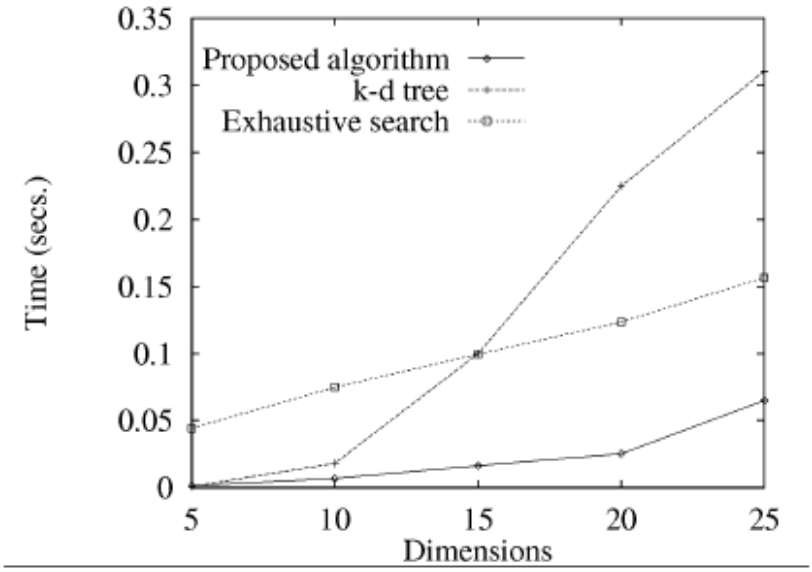

Figura 17: Comportamiento del algoritmo de Nene con una distribución normal de los datos, en comparación con la búsqueda secuencial y kd-tree

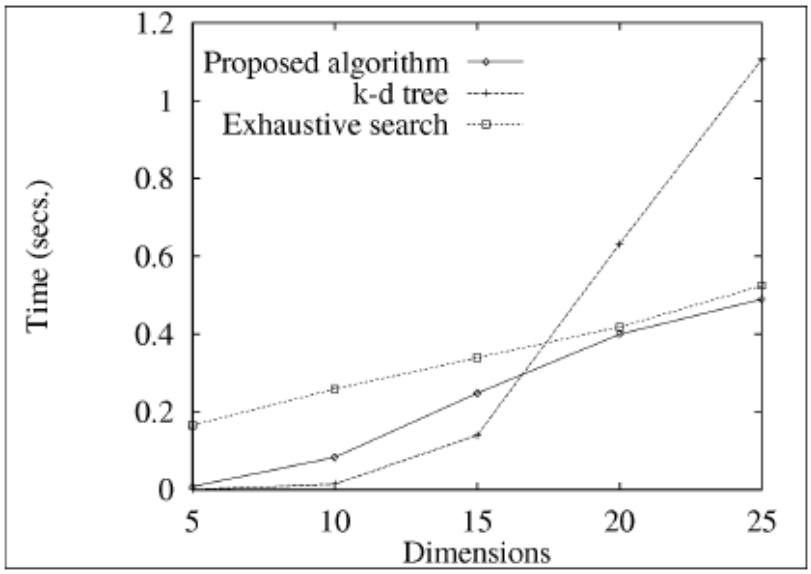

Figura 18: Comportamiento del algoritmo de Nene con una distribución uniforme de los datos, en comparación con la búsqueda secuencial y kd-tree

interna de los vectores que representan a los objetos en la memoria operativa, donde un método de acceso unidimensional podría mejorar el acceso a los elementos de un vector en el momento del cálculo de la semejanza entre objetos. Pero, para los métodos que trabajan en alta dimensionalidad, no se ha reportado su uso en problemas como los de la Minería de Texto, donde el número de dimensiones puede alcanzar las decenas de miles, por lo que es un trabajo abierto probar como se comportan a tan alta dimensionalidad.

4. Estrategias principales de paralelización

En muchas de las aplicaciones que involucran las BD multimedia se requiere procesar grandes volúmenes de datos con tiempos de respuesta acotados. Se requieren además grandes capacidades de memoria operativa y espacio de almacenamiento. Una de las soluciones prácticas al problema anterior consiste en desarrollar métodos de indexado paralelos. En la literatura aparecen propuestas de paralelización de la mayoría de los métodos de acceso secuenciales, pero además se proponen un número apreciable que son originalmente paralelos.

En este apartado se presenta un resumen de las principales propuestas de paralelización de los métodos de acceso a bases de datos multimedia que aparecen en la literatura, dividido en dos grupos: los aplicables a espacios multidimensionales y los que además presentan buenas prestaciones en espacios de muy alta dimensionalidad.

4.1. Espacios multidimensionales

4.1.1. Paralelización de datos en el R-tree

En [29] Hoel presentó una paralelización de los datos en el R-tree para determinar los polígonos cerrados en un espacio de datos de segmentos de líneas, aunque el método es extensible a otros tipos de datos.

La implementación se realizó usando el modelo SAM (*Scan and Monotonic Mapping*) sobre la arquitectura *Connection Machine*. Este modelo puede ser definido como uno o más conjuntos ordenados de procesadores los cuales permiten realizar operaciones de búsqueda sobre los elementos de manera inteligente.

El algoritmo paralelo procede como sigue: inicialmente cada línea del conjunto de datos es asociada a un procesador distinto (procesador de línea), y otro procesador es asignado al R-tree resultante (procesador de R-tree). Cada procesador de línea es asociado al procesador de R-tree. Durante el proceso de inserción se realiza una operación búsqueda hacia atrás para determinar el número de líneas asociadas al procesador de R-tree. Si este número es mayor que cierto valor M se procede a realizar una partición de las líneas de la misma forma que se realiza en el R-tree secuencial, asignando los dos nodos resultantes a dos nuevos procesadores de línea. Este algoritmo de particionamiento primero realiza un ordenamiento de todas las líneas del segmento tomando como referencia el límite izquierdo de la hipercaja de cubrimiento, que es determinada mediante otra búsqueda hacia atrás. De esta forma se va conformando un árbol de procesadores.

La complejidad de este algoritmo es O(log n) por cada etapa de construcción debido a que emplea dos ordenamientos de orden O(log n) y un número constante de operaciones de búsqueda.

Una vez terminadas las inserciones se realiza un proceso de poligonización que es el que permite determinar los polígonos cerrados. Este proceso se basa en la transmisión, a todos los procesadores de línea, de los puntos finales de cada segmento. Cuando finaliza la trasmisión todos los procesadores asignan localmente identificadores para los polígonos fronteras a la derecha y a la izquierda y mediante un proceso de mezcla sucesiva de los polígonos vecinos en los nodos padres del árbol de procesadores se logra determinar los polígonos cerrados. La complejidad del proceso de mezcla es O(n), pero el comportamiento promedio es menor.

Este algoritmo fue implementado sobre un ordenador Thinking Machines CM-2 con 16k procesadores y fue comparado contra una implementación de la versión secuencial ejecutándose sobre un ordenador Sun SPARCStation 1+, observando que la versión paralela era aproximadamente 10 veces más rápida que la secuencial.

4.1.2. Paralelización del X-tree

En 1997 Berchtold propuso [8] realizar las búsquedas del vecino más cercano, aplicando a una versión paralela del X-tree un método de distribución de los datos en varios discos externos. La idea básica de su propuesta consistió en convertir el problema de distribución de los datos en un problema equivalente a colorear grafos, haciendo coincidir los datos con los

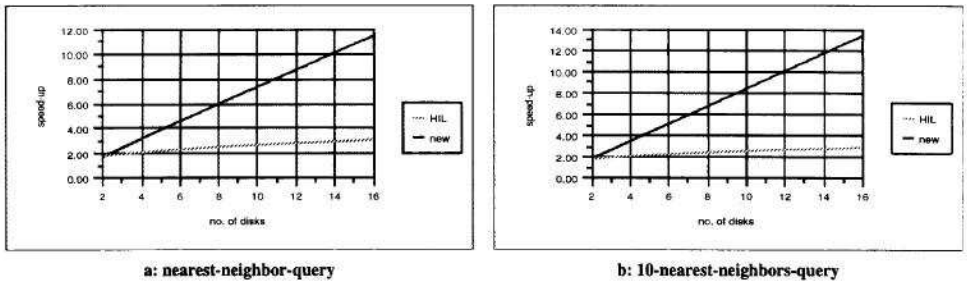

Figura 19: Comparación de las aceleraciones obtenidas utilizando el método de distribución de Berchtold con respecto al método de Hilbert, teniendo en cuenta la búsqueda del vecino más cercano (izquierda) y la búsqueda de 10 vecinos más cercanos (derecha).

vértices, las relaciones de vecindad con los ejes y los discos externos con colores. Entonces, propuso un algoritmo eficiente para resolver el problema de asignar colores al grafo.

Para mostrar la eficiencia del método propuesto lo comparó con la técnica de distribución de Hilbert, que es aplicado a problemas de espacios de datos con baja dimensionalidad.

Los experimentos fueron realizados sobre un cluster de 16 estaciones de trabajo HP710, cada una con 32 MBytes de memoria principal y gran cantidad de Mbytes de almacenamiento secundario. Uso tres tipos de datos: puntos pertenecientes al contorno de piezas industriales (con dimensiones entre 8 y 15), subcadenas de textos (con dimensión 15) y otros puntos uniformemente distribuidos (con dimensiones entre 8 y 15). Los resultados de aceleración se obtuvieron comparando el X-tree en su versión secuencial con respecto al X-tree paralelo.

Como se muestra en la figura 19 [8] las aceleraciones obtenidas con el método de distribución propuesto por Berchtold fueron superiores a las que se obtuvieron con el método de Hilbert.

4.1.3. Paralelización de datos en el εkdB-tree

En [4] se presenta una variante de paralelización del εkdB-tree para la búsqueda del vecino más cercano en espacios multidimensionales. Este tipo de árbol fue propuesto en 1997 como una estructura de indexado que escalaba mejor con respecto a la dimensionalidad que sus contemporáneas . Para la paralelización fue usada una estrategia de balance de carga basada en histogramas. En la figura 20 se muestra un ejemplo de distribución de los datos para dos procesadores. Cada procesador almacena la estructura del árbol, pero las hojas son distribuidas entre estos (las hojas faltantes en cada procesador son indicadas con líneas discontinuas en la figura), por lo que en cada procesador se almacena sólo un subconjunto de los datos indexados.

En el artículo se proponen dos algoritmos paralelos para realizar la búsqueda del vecino más cercano: PW y PQ. Ambos están divididos en cuatro fases principales: fase de particionamiento, en la que el espacio de datos es dividido en regiones disjuntas; cada región es asignada a un procesador y en la segunda fase se construyen árboles de indexado εkdB-tree locales; en la fase 3, cada procesador determina con cuales procesadores debe intercambiar

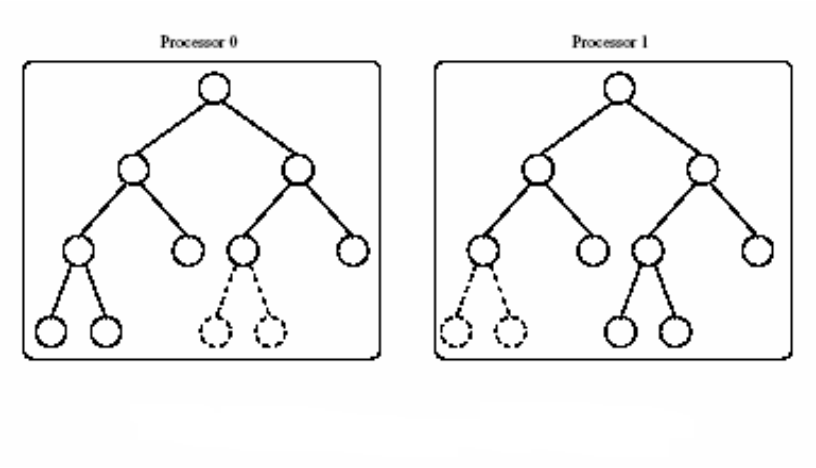

Figura 20: Distribución de los datos en el εkdB-tree para el caso de dos procesadores

datos y las subregiones que deben ser comunicadas. Posteriormente la búsqueda se realiza en cada región de forma independiente y se obtiene la solución, eliminando las repeticiones.

Los algoritmos fueron implementados sobre una IBM SP-2 con 16 procesadores utilizando la librería de comunicación MPI. Cada procesador era un RS/6000 a 66.7 MHz, con 128 MBytes de memoria RAM por procesador, y la versión 4 del sistema operativo AIX.

Para las pruebas se utilizaron juegos datos con distintos tipos de distribuciones: uniforme, gausiana y dos tipos distintos de datos mixtos. Los resultados demostraron que las propuestas escalan con respecto al número de procesadores utilizados, pero con el PW se obtuvieron mayores aceleraciones.

4.1.4. Paralelización del M-tree

En 1998 Alpkocak et. al presentan en [3] una versión paralela del M-tree usando un mecanismo de distribución de los objetos que permite paralelizar tanto el acceso a la información en disco como el procesamiento en la memoria principal. Alpkoca define el problema del distribución como una distribución adecuada de los objetos entre los discos y los procesadores, que permita realizar una consulta distribuida de la forma más homogénea posible, de forma tal que se pueda explotar al máximo la paralelización de los cálculos y las comunicaciones.

El método propuesto tiene en cuenta tanto la proximidad mutua de los objetos como el balance de carga de los procesadores.

Para evaluar el algoritmo de distribución propuesto se utilizó una base de datos que contenía 1000 histogramas de imágenes a color con 32 bits en el espacio de color HSV. Se utilizó la distancia Euclidiana para medir la semejanza. El método propuesto es 7 veces más rápido que la consulta de tipo k-NN, sobre un cluster de procesadores.

En [70] Zezula et. al presentan cuatro estrategias diferentes de distribución de los objetos para la paralelización del M-tree obteniendo niveles de aceleración y escalabilidad altos. Estas

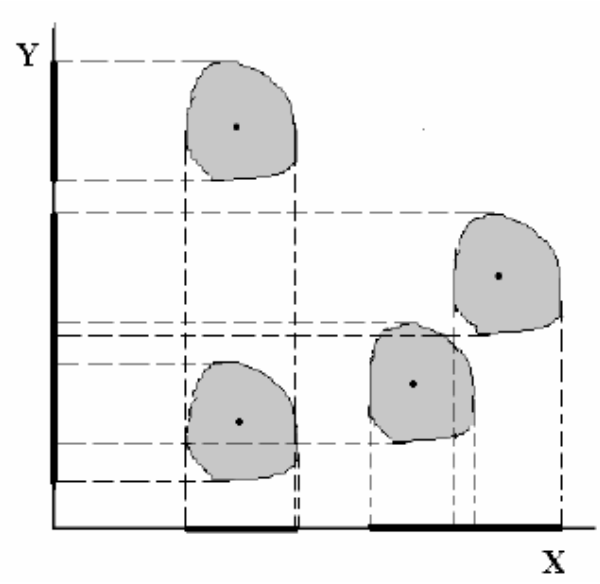

Figura 21: Agrupamiento y proyección de los objetos en el PN-tree

estrategias fueron evaluadas variando el número de procesadores (1-20), el número de discos (1-20) y la dimensión de los datos (2-45), con un juego de 20000 objetos sintetizados y realizando la búsqueda de los 10 vecinos más cercanos. Se utilizó para calcular la semejanza entre objetos la distancia euclidiana. Las pruebas se realizaron sobre un ordenador con múltiples procesadores, donde uno de estos contenía la estructura del M-tree y el resto se dedicaron a la medición de las distancias; y un conjunto de discos paralelos independientes utilizados para almacenar los nodos del árbol.

Las conclusiones más importantes de este trabajo fueron: el nivel de escalabilidad y aceleración obtenidos fueron similares a los reportados para el X-tree paralelo [11] en [8]; los métodos de distribución basados en la semejanza reportan buenas prestaciones cuando el número de discos es menor que el número de nodos; y finalmente, si se cuenta con un elevado número de discos, es más aconsejable utilizar el método de distribución aleatorio.

4.1.5. PN-tree

En 2001 Saad presenta una nueva estructura de indexado para espacios multidimensionales, paralela por naturaleza, el PN-tree [2], que se aprovecha tanto del número de procesadores como de la dimensionalidad del espacio de representación de los objetos. Esta estructura está basada en el B*-tree. La idea central de la paralelización se basa en agrupar los objetos en nodos que luego son proyectados sobre cada dimensión del espacio con el objetivo de obtener diferentes vistas de este (figura 21 [2]).

Finalmente, cada dimensión es indexada de forma separada como espacios unidimensionales. Los resultados experimentales realizados por su autor demuestran que el uso del PN-tree reduce significativamente el número de accesos a disco durante una operación de búsqueda, además de que muestra prestaciones superiores al Hybrid-tree para BDs de gran

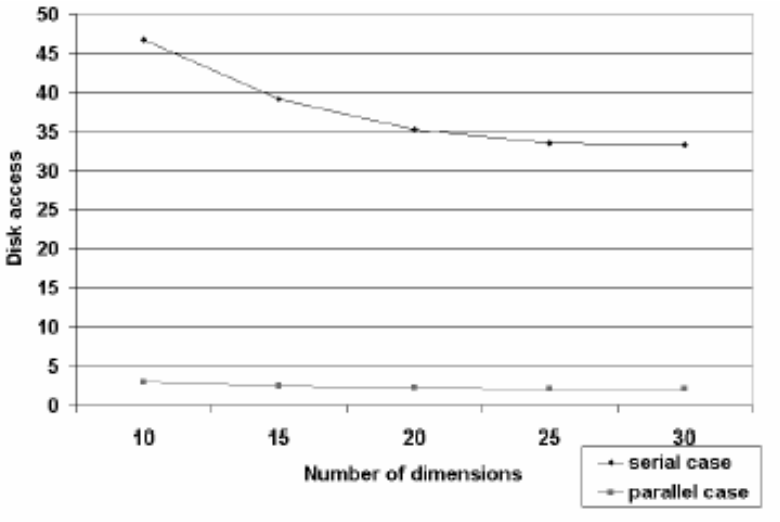

Figura 22: Efecto de la dimensionalidad sobre el PN-tree

tamaño. En la figura 22 [2] se muestra el efecto de la dimensionalidad sobre esta estructura y en la figura 23 [2] el efecto producido por el número de procesadores.

Estos resultados fueron obtenidos sobre un multiprocesador con 16 procesadores.

Como se aprecia el número de accesos a disco disminuye a medida que se incrementa el número de dimensiones del espacio de datos, y también cuando se incrementa el número de procesadores.

4.2. Propuestas de paralelización de estructuras de indexado para espacios de muy alta dimensionalidad

El método VA-file también ha sido objeto de paralelización, y en [65] Weber presenta los resultados obtenidos usando dos configuraciones distintas. La primera asignando varios discos a un sólo procesador, y la segunda, usando un cluster de PCs.

En el primer caso se proponen dos variantes, la primera nombrad*a VA-SSA paralelo* (*SSA*-algoritmo simple de búsqueda), donde se reduce el coste de entrada y salida (lecturas en disco), almacenando los vectores y sus aproximaciones en discos distintos. Se crea un hilo para cada disco y un hilo para procesar las aproximaciones. Esta solución es escalable, y en la figura 24 [65] se muestran los resultados presentados por Weber con respecto a la reducción en el tiempo de búsqueda y el incremento de aceleración en función del número de discos.

La segunda variante, *VA-NOA paralelo (NOA*-algoritmo cercano óptimo), se divide en dos fases, cada una de las cuales lee o los datos originales o las aproximaciones, por lo que cada fichero de datos puede ser distribuido sobre el conjunto de discos con un mayor nivel de paralelismo. En la figura 25 se muestran los resultados de esta variante en función del número de discos.

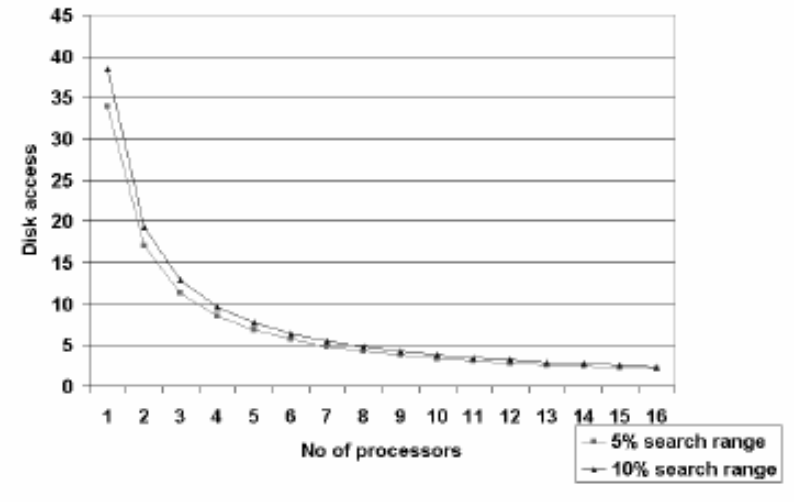

Figura 23: Efecto del número de procesadores sobre el PN-tree

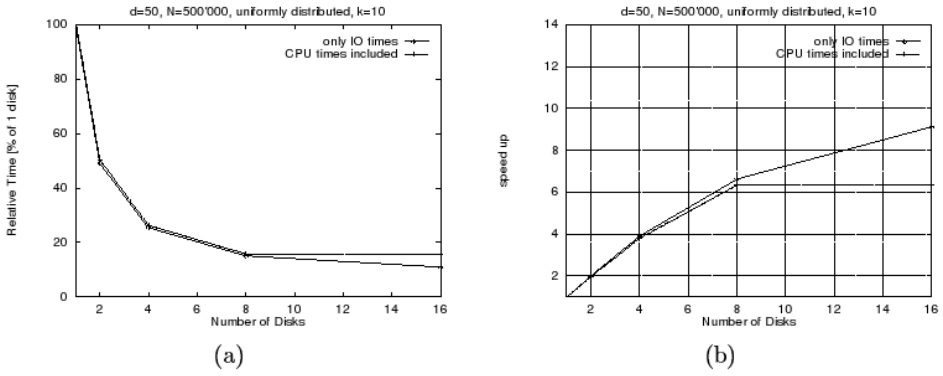

Figura 24: Reducción del tiempo de búsqueda (a); e incremento de aceleración del *Parallel VA-SSA* en función del número de discos.

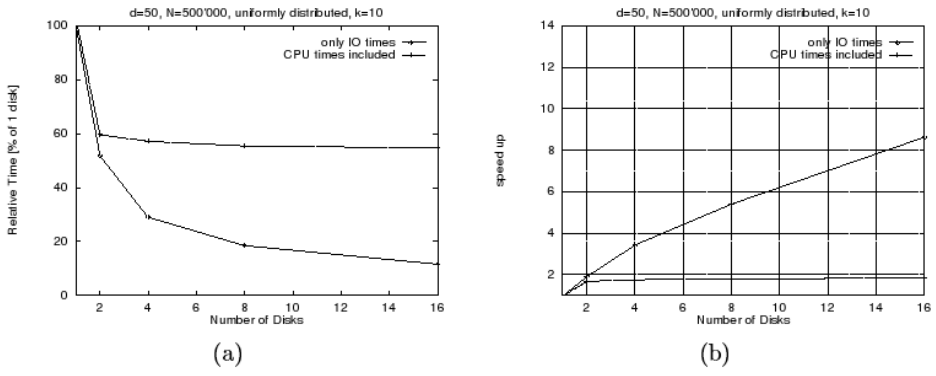

Figura 25: Reducción del tiempo de búsqueda (a); e incremento de prestaciones del *Parallel VA-NOA* en función del número de discos.

El segundo caso, denominado *VA-File distribuido*, aprovecha de los clusters de PCs tanto la capacidad de cálculo de los ordenadores como la capacidad de los discos. La paralelización se basa en el hecho de que la consulta original puede ser descompuesta en varias subconsultas más pequeñas. Estas subconsultas son enviadas a los nodos que participan en el cluster, donde existen réplicas de los índices. Finalmente los resultados de las subconsultas son combinados para obtener la respuesta a la consulta original.

Esta última variante ofrece dos ventajas adicionales sobre la paralelización basada en múltiples discos. La primera consiste en el incremento de la disponibilidad, debido a la replicación de los índices, y la segunda en el decremento del tiempo total de respuesta debido a que varias consultas pueden ser procesadas simultáneamente.

Estos resultados fueron obtenidos con implementaciones sobre una Sun UltraSPARC con 128 MB de RAM. Los datos (50000 puntos de 50 dimensiones distribuidos uniformemente) fueron almacenados en el disco local.

Para el caso del IQ-tree no tenemos conocimiento de que se haya propuesto alguna paralelización.

En resumen, hemos presentado las principales técnicas de paralelización de los métodos de acceso a BD de multimedia descritas en la literatura. Las distintas estrategias persiguen la distribución óptima de los datos a través de un conjunto de discos externos o la división de las consultas en subconsultas para aprovechar la potencia de cálculo de los procesadores. En todos los casos con mayor o menor éxito se ha logrado mejorar las prestaciones de las estructuras originales, pero los resultados mostrados han sido obtenidos sobre conjuntos de datos que no sobrepasan las 50 dimensiones.

5. Conclusiones

El objetivo central de este informe consistió en comparar los métodos principales de indexado para el acceso a las BD multimedia existentes en la literatura, especialmente en cuanto a su comportamiento con respecto a la dimensionalidad del espacio de datos, ya que tenemos especial interés en aplicarlos a la Minería de Textos.

Hemos presentado los conceptos básicos referentes a estos métodos, hemos descrito las principales aproximaciones estudiadas en la literatura y mostramos que la mayoría de estas han sido desarrolladas para espacios de datos de baja o media dimensionalidad.

La mayoría de estos métodos presentan malas prestaciones cuando el número de dimensiones es elevado, como ocurre, por ejemplo, cuando se trabaja con colecciones de textos, donde las dimensiones pueden llegar a ser de decenas de miles. Para estos casos se han diseñado estructuras específicas que son inmunes a la alta dimensionalidad, y que lo logran usando métodos de acceso con muy alta selectividad. En algunos de los casos son incluso considerados meros métodos de compactación en vez de estructuras de indexado.

Hemos presentado además, las principales variantes de paralelización usadas con el objetivo de mejorar las prestaciones finales de los mismos.

De este estudio podemos concluir que:

- De las funciones estudiadas para el cálculo de las semejanzas entre objetos, el dual de la función coseno es suficiente para nuestros propósitos.
- El tipo de consulta apropiado para nuestro caso es el de semejanza, o sea, determinar los k-vecinos más cercanos y los distantes a menos que un radio dado.
- De los métodos de acceso estudiados, pueden aplicarse los diseñados para trabajar en alta dimensionalidad, como son el VA-file, IQ-tree y el Método de Nene, que tienen una selectividad muy alta y presentan mayores prestaciones a medida que crece la dimensionalidad del problema. O podrían desarrollarse nuevos métodos que reúnan las mejores características de los actuales y obtengan mejores prestaciones.
- Las prestaciones de los métodos para alta dimensionalidad han sido reportadas sólo para un número de dimensiones relativamente alto, pero nunca para dimensiones tan altas, como las presentes en la Minería de Textos, lo que hace interesante el estudio del comportamiento de estos en este tipo de aplicación.
- Las paralelizaciones desarrolladas hasta la fecha, mejoran varias veces las prestaciones de los métodos secuenciales, por lo que es perfectamente factible su uso en el problema de la Minería de Textos, pero no todos los métodos han sido paralelizados, por lo que queda un camino abierto para su desarrollo.
- Finalmente, otros métodos de acceso, como los unidimensionales, aunque no pueden ser usados para indexar los objetos de alta dimensionalidad, pueden utilizarse para mejorar el acceso a los elementos que conforman los vectores que los representan, con el objetivo de optimizar el cálculo de la semejanza entre estos.

Referencias

[1] H.-W. Six A. Henrich and P. Widmayer. The lsd tree: Spatial access to multidimensional point and nonpoint objects. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, August 22-25 1989.

[2] M. H. Ali, A. A. Saad, and M. A. Ismail. The pn-tree: a parallel and distributed multidimensional index. *Distrib. Parallel Databases*, 17(2):111–133, 2005.

[3] Adil Alpkocak, Taner Danisman, and Tuba Ulker. A parallel similarity search in high dimensional metric space using m-tree. In *IWCC '01: Proceedings of the NATO Advanced Research Workshop on Advanced Environments, Tools, and Applications for Cluster Computing-Revised Papers*, pages 166–171, London, UK, 2002. Springer-Verlag.

[4] Alsabti, Ranka, and Singh. An efficient parallel algorithm for high dimensional similarity join. In *IPPS: 11th International Parallel Processing Symposium*. IEEE Computer Society Press, 1998.

[5] Lars Arge, Klaus Hinrichs, Jan Vahrenhold, and Jeffrey Scott Vitter. Efficient bulk operations on dynamic r-trees. *Algorithmica*, 33(1):104–128, 2002.

[6] R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica 1*, pages 173–189, 1972.

[7] R.Sacks-Davis B.C.Ooi and K.J.McDonell. Spatial k-dtree: An indexing mechanism for spatial databases. In *IEEE COMPSAC Conference*, 1987.

[8] Stefan Berchtold, Christian Bohm, Bernhard Braunuller, Daniel A. Keim, and Hans-Peter Kriegel. Fast parallel similarity search in multimedia databases. *SIGMOD Rec.*, 26(2):1–12, 1997.

[9] Stefan Berchtold, Christian Bohm, H. V. Jagadish, Hans-Peter Kriegel, and Jorg Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, pages 577–588, San Diego, USA, 2000.

[10] Stefan Berchtold, Christian Bohm, H. V. Jagadish, Hans-Peter Kriegel, and Jorg Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *ICDE*, pages 577–588, 2000.

[11] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *Proceedings of the 22nd International Conference on Very Large Databases*, pages 28–39, San Francisco, U.S.A., 1996. Morgan Kaufmann Publishers.

[12] Zhou Bing, Shen Jun-yi, and Peng Qin-ke. A content-based parallel image retrieval system on cluster architectures. *Wuhan University Journal of Natural Sciences*, 9(5):665–670, 2004.

[13] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM Press.

[14] S. Brakatsoulas, D. Pfoser, and Y. Theodoridis. Revisiting r-tree construction principles, 2002.

[15] Kaushik Chakrabarti and Sharad Mehrotra. The hybrid tree: An index structure for high dimensional feature spaces. In *ICDE*, pages 440–447, 1999.

[16] Seonghun Cho and Sartaj Sahni. A new weight balanced binary search tree. *International Journal of Foundations of Computer Science*, 11(3):485–513, 2000.

[17] P. Ciaccia and M. Patella. Bulk loading the m-tree. In *Proceedings of the 9th Australasian Database Conference (ADC'98)*, pages 15–26, Perth, Australia, February 1998.

[18] Paolo Ciaccia, Marco Patella, Fausto Rabitti, and Pavel Zezula. Indexing metric spaces with m-tree. In *Sistemi Evolui per Basi di Dati*, pages 67–86, 1997.

[19] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *The VLDB Journal*, pages 426–435, 1997.

[20] Mark de Berg, Joachim Gudmundsson, Mikael Hammar, and Mark H. Overmars. On r-trees with low stabbing number. In *European Symposium on Algorithms*, pages 167–178, 2000.

[21] Sampath Deegalla and Henrik Bostrom. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*, pages 245–250, Washington, DC, USA, 2006. IEEE Computer Society.

[22] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[23] J.H. Friedman, F. Baskett, and L.J. Shustek. An algorithm for finding nearest neighbors. *IEEE Trans. Computers*, pages 1000–1006, 1975.

[24] Ada Wai-Chee Fu, Polly Mei shuen Chan, Yin-Ling Cheung, and Yiu Sang Moon. Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. *VLDB Journal*, 9(2):154–173, 2000.

[25] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, pages 518–529, Edinburgh, Scotland, September 1999.

[26] Oliver Gunther. The design of the cell tree: An object-oriented index structure for geometric databases. In *Proceedings of the Fifth International Conference on Data Engineering*, pages 598–605, Los Angeles, California, USA, February 1989. IEEE Computer Society.

[27] A. Guttman. R-tree: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Int. Conf. Management of Data*, pages 47–54, 1984.

[28] Gisli R. Hjaltason and Hanan Samet. Ranking in spatial databases. In *Symposium on Large Spatial Databases*, pages 83–95, 1995.

[29] Erik G. Hoel and Hanan Samet. Data-parallel R-tree algorithms. In *Proceedings of the 1993 International Conference on Parallel Processing*, volume III - Algorithms Applications, pages III–47–III–50, Boca Raton, FL, 1993. CRC Press.

[30] I.Galperin and R.L.Rivest. Scapegoat trees. *Proceedings of SODA 1993*, pages 165–174, 1993.

[31] J.L.Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[32] J.Nievergelt and E.M.Reingold. Binary search trees of bounded balance. *Proceedings of the fourth annual ACM symposium on Theory of Computing*, (3):137–142, 1972.

[33] J.T.Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 10–18, Ann Arbor, MI, 1981.

[34] M. R. K. and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In M. A. Nascimento, M. T. Özsu, D. Kossman, R. J. Miller, J. A. Blakely, and K. B. Schiefer, editors, *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 840–851, Toronto, Canada, September 2004.

[35] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. In *CIKM*, pages 490–499, 1993.

[36] George Kollios, Vassilis J. Tsotras, Dimitrios Gunopulos, Alex Delis, and Marios Hadjieleftheriou. Indexing animated objects using spatiotemporal access methods. *Knowledge and Data Engineering*, 13(5):758–777, 2001.

[37] Hans-Peter Kriegel, Thomas Brinkhoff, and Ralf Schneider. Efficient spatial query processing in geographic database systems. *Data Engineering Bulletin*, 16(3):10–15, 1993.

[38] Scott T. Leutenegger, Jeffrey M. Edgington, and Mario A. Lopez. STR: A simple and efficient algorithm for R-tree packing. Technical Report TR-97-14, 1997.

[39] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. The TV-tree: An index structure for high-dimensional data. *VLDB Journal: Very Large Data Bases*, 3(4):517–542, 1994.

[40] M.A.Nascimento and J.R.O.Silva. Towards historical r-trees. *ACM SAC*, 1998.

[41] Yannis Manolopoulos, Alexandros Nanopoulos, ApostolosÑ. Papadopoulos, and Yannis Theodoridis. R-trees have grown everywhere, 2003.

[42] A. Mhatre, S. Chikkerur, and Govindajaru. Indexing biometric databases using pyramid technique. *AVBPA 2005*, pages 841–849, 2005.

[43] Nicolas Moene-Loccoz. High-dimensional access methods for efficient similarity queries. Technical Report 05.05, Computer Vision Group. University of Geneva. Switzerland, 2005.

[44] R. Schneider N. Beckmann, H.-P. Kriegel and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM-SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, 1990.

[45] Mario A. Nascimento, Jefferson R. O. Silva, and Yannis Theodoridis. Evaluation of access structures for discretely moving points. In *Spatio-Temporal Database Management*, pages 171–188, 1999.

[46] Sameer A. Nene and Shree K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1997.

[47] Yutaka Ohsawa and Masao Sakauchi. A new tree type data structure with homogeneous nodes suitable for a very large spatial database. In *Proceedings of the Sixth International Conference on Data Engineering*, pages 296–303, Washington, DC, USA, 1990. IEEE Computer Society.

[48] Bernd-Uwe Pagel, Flip Korn, and Christos Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *ICDE*, pages 589–598, 2000.

[49] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In *The VLDB Journal*, pages 395–406, 2000.

[50] P.L. Lin P.W. Huang and H.Y. Lin. Optimizing storage utilization in r-tree dynamic index structure for spatial databases. *Journal of Systems and Software*, 55:291–299, 2001.

[51] H.-J. Schek R. Webwer and S. Blott. A quantitative analisys and performance study for similarity-search methods in high-dimensional spaces. In *24th International Conference on Very Large Data Bases (VLDB'98)*, pages 194–205, New York, USA, 1998.

[52] Nick Roussopoulos and Daniel Leifker. Direct spatial search on pictorial databases using packed r-trees. In *SIGMOD '85: Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, pages 17–31, Austin, Texas, United States, 1985. ACM Press.

[53] C. Bohm S. Berchtold and H.-P Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. 1998.

[54] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. The A-tree: An index structure for high-dimensional spaces using relative approximation. In *Proc. of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 516–526, Cairo, September 2000.

[55] S. Saltenis and C. Jensen. R-tree based indexing of general spatiotemporal data, 1999.

[56] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD Conference*, pages 331–342, 2000.

[57] Ralf Schneider and Hans-Peter Kriegel. The tr*-tree: A new representation of polygonal objects supporting spatial queries and operations. In *CG '91: Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications*, pages 249–263, London, UK, 1991. Springer-Verlag.

[58] Bernhard Seeger and Hans-Peter Kriegel. The buddy-tree: An efficient and robust access method for spatial data base systems. In *16th VLDB*, pages 507–518, 1987.

[59] Thomas Seidl and Hans-Peter Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD Conference*, pages 154–165, 1998.

[60] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The r -tree: A dynamic index for multi-dimensional objects. In *The VLDB Journal*, pages 507–518, 1987.

[61] Yufei Tao and Dimitris Papadias. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *The VLDB Journal*, pages 431–440, 2001.

[62] Yannis Theodoridis, Michalis Vazirgiannis, and Timos K. Sellis. Spatio-temporal indexing for large multimedia applications. In *International Conference on Multimedia Computing and Systems*, pages 441–448, 1996.

[63] K. Richta V. Snasel, J. Pokorn. Pivoting m-tree: A metric access method for efficient similarity search. Technical report, Technical University of Ostrava, Dept. of Computer Science, 2004.

[64] R. Weber and S. Blott. An approximation based data structure for similarity search, 1997.

[65] Roger Weber. Parallel va-file. Technical Report 25, ESPRIT project HERMES (project no. 9141), 1997.

[66] Jeffrey Xu Yu2 Xiangmin Zhou1, Guoren Wang1 and Ge Yu1. M+-tree: A new dynamical multidimensional index for metric spaces. In Xiaofang Zhou and Eds Klaus-Dieter Schewe, editors, *Proceedings of the Conferences in Research and Practice in Information Technology*, Adelaide, Australia, 2003. Australian Computer Society, Inc.

[67] Li Yang. Distance-preserving projection of high-dimensional data for nonlinear dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1243–1246, 2004.

[68] Q. Yang, A.Vellaikal, and S. Dao. Mb + -tree: A new index structure for multimedia databases. In *Proc. of the International Workshop on Multi-Media Database*, pages 151–158, 1995.

[69] Pavel Zezula, Pasquale Savino, Giuseppe Amato, and Fausto Rabitti. Approximate similarity retrieval with m-trees. volume 7, pages 275–293, 1998.

[70] Pavel Zezula, Pasquale Savino, Fausto Rabitti, Giuseppe Amato, and Paolo Ciaccia. Processing m-trees with parallel resources. In *RIDE*, pages 147–154, 1998.