Un método de acceso aproximado para muy alta dimensionalidad y su paralelización

Fernando Artigas Fuentes[1] José Manuel Badía Contelles[2] y Reynaldo Gil García[3]

Resumen— **En este trabajo, proponemos un nuevo método de acceso aproximado para espacios de muy alta dimensionalidad y especialmente aquellos con muy alta dispersión de los datos. Está basado en el uso de múltiples grafos como estructura de indexado y un algoritmo de búsqueda para obtener de un repositorio de objetos, representados como vectores, aquellos más relacionados a otros objetos usados como consultas. Aunque este método es aproximado, muestra un nivel bajo de error. Tiene además un coste temporal muy bajo durante las consultas y está especialmente diseñado para ser ejecutado en paralelo de forma simple y eficiente. Mostramos versiones paralelas tanto para la generación de la estructura de indexado como para el algoritmo de búsqueda. Se ha paralelizado usando la interfaz OpenMP. Hemos evaluado nuestra propuesta sobre espacios de representación de miles dimensiones obteniendo prestaciones paralelas cercanas al óptimo.**

Palabras clave— **métodos de acceso, computación paralela, búsqueda aproximada, grafos.**

I. Introducción

PARA recuperar información, es muy común que se utilice un método de acceso. Este tiene el objetivo de organizar los objetos de un repositorio de forma tal que durante las consultas solo sea necesario acceder a una porción del mismo. Cuanto menor sea la porción visitada para cada consulta, más eficiente resulta el método de acceso.

La mayoría de los datos que se manejan en la actualidad son descritos mediante un gran número de características, que para los métodos de acceso se convierten en dimensiones del espacio de representación de los datos.

Existe en la literatura un conjunto amplio de métodos de acceso capaces de indexar espacios multidimensionales, pero la mayoría sufre un problema conocido como "Maldición de la dimensionalidad" [1]. Este provoca que a medida que aumenta la dimensionalidad del espacio de representación de los datos los resultados de la búsqueda se van degradando hasta convertirse en iguales o peores que para el caso de búsqueda exhaustiva.

Debido a que todos los métodos exactos sufren el problema antes descrito, para el caso de espacios de muy alta dimensionalidad, se han presentado en la literatura nuevos métodos que obtienen soluciones aproximadas. Estos métodos utilizan varias técnicas que consisten en relajar las condiciones tanto de la construcción de las estructuras de indexado como de las estrategias de búsqueda. Algunas de las técnicas usadas son la reducción de la dimensionalidad del espacio de búsqueda, seleccionando un subconjunto de características más representativas de los objetos, hasta valores más manejables por los métodos ya existentes [2], y la sustitución de los objetos originales por otros menos complejos que mantengan aproximadamente las mismas relaciones de vecindad que los originales [6].

Otras propuestas son eficientes manejando un gran número de dimensiones, como es el caso del VA-file y los derivados de este [3], [4], [5]. Sin embargo, estos métodos se vuelven ineficientes cuando a la alta dimensionalidad de los datos tratados se suma una alta dispersión de los mismos.

En este trabajo proponemos un nuevo método de acceso que es poco afectado por ambos problemas: la alta dimensionalidad y la alta dispersión de los datos. El mismo está basado en el uso de múltiples grafos como estructura de indexado y un algoritmo de búsqueda que, aunque aproximado, permite obtener un elevado porcentaje de soluciones exactas con un coste temporal muy bajo.

Además, la estructura de datos se ha diseñado con el fin de poder construirla y utilizarla en paralelo de forma sencilla y eficiente. Presentamos además una versión paralela de este método, mediante el uso de memoria compartida y la interfaz OpenMP.

El resto del trabajo está organizado como sigue: en la sección 2 se trata con más detalles las propiedades de los métodos de acceso, en la sección 3 explicamos nuestra propuesta y su paralelización, en la sección 4 describimos los experimentos realizados y los resultados obtenidos. Finalmente en la sección 5 mostramos las conclusiones de este trabajo.

II. Métodos de acceso

Un método de acceso está formado por una estructura de indexado, que organiza de alguna manera los objetos de un repositorio, y una estrategia de búsqueda que debe recorrer los objetos indexados de la manera más eficiente.

Un método de acceso necesita además de una medida de comparación entre los objetos que exprese la proximidad entre estos. Esta puede ser una semejanza, que da una medida de en cuanto se parecen dos objetos entre sí; o una diferencia, que determina en cuanto se distinguen. Por lo general como medida de comparación se usa una función de distancia, que además de expresar lo cercano que están los objetos entre sí, define un espacio métrico.

Una función distancia puede expresarse como:
$dist : \Omega_d x \Omega_d \rightarrow \Re^+$

[1]CERPAMID, Univ. Oriente, Cuba, e-mail: artigas@cerpamid.co.cu.
[2]Dpto. de Ingeniería y Ciencias de los Computadores, Univ. Jaume I, e-mail: badia@icc.uji.es.
[3]CERPAMID, Univ. Oriente, Cuba, e-mail: gil@cerpamid.co.cu.

donde Ω_d es el espacio de representación de los datos y d la dimensionalidad del mismo.

Una distancia debe cumplir los siguientes requisitos:

Reflexividad: $\forall x \in \Omega_d$, $dist(x,x) = 0$

Simetría: $\forall x, y \in \Omega_d$, $dist(x,y) = dist(y,x)$

Desigualdad triangular: $\forall x, y, z \in \Omega_d, dist(x,y) \leq dist(x,z) + dist(z,y)$

Si el espacio de representación usado no es un espacio métrico, entonces es suficiente con que la medida de comparación cumpla con los dos primeros.

Debido a que las medidas pueden ser diversas, cuando comparemos dos objetos, nos referiremos simplemente a proximidad, y la denotaremos por Ψ, por lo que se cumple que:

$\forall x, y, z \in \Omega_d$, $\Psi(x,y) < \Psi(x,z) \rightarrow sem(x,y) > sem(x,z)$

$\forall x, y, z \in \Omega_d$, $\Psi(x,y) < \Psi(x,z) \rightarrow dif(x,y) < dif(x,z)$

donde sem y dif expresan que la medida usada es una semejanza o una diferencia, respectivamente.

Las consultas pueden ser exactas o aproximadas. En el primer caso deben ser obtenidos, si existen, todos los objetos pertenecientes al repositorio de datos que cumplan con que su proximidad con la consulta sea máxima, esté dentro de un rango determinado de proximidad, o pertenezca al conjunto de los k elementos más próximos a la misma. Esta última condición por lo general se expresa como obtener los k vecinos más cercanos (kNN) a la consulta.

Debido a que los métodos de acceso que usan estrategias de búsqueda exactas fallan para el caso de espacios de datos de muy alta dimensionalidad, es deseable el uso de estrategias aproximadas. No obstante, estas deben garantizar que los datos obtenidos tengan una alta probabilidad de pertenecer a las soluciones reales, o que el nivel de error sea limitado.

De manera general, de un método de acceso se espera que sea capaz de indexar los datos de un repositorio en el menor espacio posible; que tengan una alta selectividad, o lo que es lo mismo que sean capaces de descartar el mayor número posible de datos que no solapen con las soluciones a las consultas; y que sean escalables con la dimensionalidad.

En la siguiente sección presentamos nuestra propuesta: un método de acceso aproximado, que usa como estructura de indexado un conjunto de grafos y cuya estrategia de búsqueda tiene una alta selectividad. Este método escala bien con la dimensionalidad del espacio y tiene un coste temporal muy bajo durante las consultas. Por otra parte las soluciones obtenidas tienen un nivel de error muy bajo, obteniéndose, en los casos estudiados, las respuestas exactas para más del 90% de los casos.

III. Método de acceso propuesto

A. Estructura de indexado

La estructura propuesta indexa los objetos del repositorio en N grafos conexos no orientados que cumplen las siguientes condiciones:

1. Se usan las representaciones vectoriales de los objetos.
2. Por cada dimensión se especifica un valor, que representa el peso de la misma.
3. El repositorio es dividido en N subconjuntos disjuntos, no vacíos, que lo cubren completamente.
4. Cada subconjunto es indexado por un solo grafo.
5. Dentro de un grafo, cada vértice se corresponde con un solo objeto del subconjunto correspondiente.
6. Cada vértice está conectado directamente con al menos una mínima cantidad (θ) de otros vértices, que representan a los objetos más próximos a este.
7. En cada grafo, los vértices que satisfacen ciertas condiciones son seleccionados como puntos de entrada al mismo durante las búsquedas.

Una vez convertidos los objetos originales en su representación vectorial y repartidos en un número predeterminado de subconjuntos, se procede a generar los N grafos, aplicando a cada subconjunto el Algoritmo 1.

Cada uno de estos grafos contiene su propio conjunto de vértices de entrada E_p. Estos vértices son usados durante las consultas como puntos de pártida para la estrategía de búsqueda.

B. Estrategia de búsqueda

La estrategia de búsqueda que se sigue es muy simple (Algoritmo 2): las consultas son evaluadas sobre cada grafo mediante el Algoritmo 3, obteniéndose una solución parcial para cada uno de estos. A partir de estas soluciones parciales se obtiene la solución final. Para esto se crea un conjunto con las soluciones aportadas por cada grafo y de este conjunto se seleccionan los k elementos con máxima proximidad a la consulta.

A pesar de que mostraremos que la estrategia de búsqueda propuesta garantiza costes temporales muy bajos durante las consultas, hay situaciones en las que es necesario procesar un gran flujo de consultas. Por esta razón, es deseable contar con una versión paralela del método propuesto que permita aprovechar al máximo las arquitecturas actuales que por lo general cuentan con más de un procesador.

A continuación describimos la paralelización del método de acceso propuesto.

C. Paralelización de la generación de la estructura de indexado

La generación de cada grafo G_p con el Algorithm 1 es completamente independiente del resto, por lo que la paralelización de este proceso es inmediata. Basta con aplicar un esquema completamente paralelo, asignando cada subconjunto de datos a un procesador distinto.

La asignación de los objetos a cada subconjunto se hace seleccionándolos de manera aleatoria. Debido a que tratamos con objetos cuyos vectores son

Algoritmo 1 GenerarGrafo(O_p, θ)

Entrada: O_p: subconjunto de vectores a indexar, θ: número mínimo de vecinos directos de cada vértice en el grafo final.

Salida: G_p: grafo que indexa a los vectores de entrada.

Etapa1. Cálculo del centroide del subconjunto.

Se obtiene un nuevo vector como resultado de sumar todos los vectores del conjunto dimensión por dimensión. El vector resultante tiene valor mayor que cero en todas las dimensiones del espacio de representación de este subconjunto.

Etapa 2. Descomposición del subconjunto en una lista ordenada de subconjuntos por su proximidad con el centroide.

A partir de los elementos del subconjunto se obtiene una lista ordenada en orden decreciente de proximidad con el centroide del subconjunto.

Esta lista es dividida en un número predefinido de sublistas, que quedarán ordenadas por la proximidad de sus elementos con el centroide.

Etapa 3. Construcción del grafo conexo.

De los elementos de la primera sublista, se calcula el par de elementos más próximos entre sí. Estos elementos forman la primera arista del grafo.

Del resto de los elementos de la sublista se calcula el elemento que al ser conectado a un par de vértices de cualquier arista del grafo provoca el menor crecimiento en área de esta.

Este proceso se repite hasta que todos los elementos de la sublista son conectados al grafo.

Los dos pasos anteriores son repetidos para el resto de las sublistas, hasta que todos los elementos de la lista original estén conectados en el grafo.

Etapa 4. Completar el grafo hasta que cada vértice tenga θ vecinos.

En esta etapa se verifica que cada vértice forme parte como mínimo de una cantidad predefinida de aristas. En caso que se detecte un vértice que no cumpla esta condición el mismo es conectado con nuevos vecinos más próximos hasta que se cumpla la condición.

Etapa 5. Calcular los vértices de entrada al grafo.

Se determinan aquellos vértices que cumplen que todos los vértices con los que conforman aristas están más próximos que estos al centroide del subconjunto, y aquellos vértices que cumplen que todos los vértices con los que conforman aristas están menos próximos que estos al centroide del subconjunto.

Finalmente se genera una estructura que contiene al conjunto de vértices (O_p), al conjunto de aristas (A_p) y al conjunto de los vértices identificados como de entrada al grafo (E_p).

return $G_p = (O_p, A_p, E_p)$.

Algoritmo 2 BuscarKPróximos(R, G, q, k)

Entrada: R, el repositorio de objetos iniciales; $G = G_1, G_2, ..., G_N$, la lista de los grafos obtenidos mediante el Algoritmo 1; q, el objeto de consulta; y k, el número de soluciones requeridas.

Salida: Los k objetos de R más próximos a q según Ψ.

Sea $\overrightarrow{v} = \zeta(q)$ la representación vectorial de q.

Sean $G_i = G[i]$, $O_i = G_i[1]$,$A_i = G_i[2]$ y $E_i = G_i[3]$.

Sean $S^{'} = \emptyset$ y $N = |G|$.

for $i = 1 \rightarrow N$ **do**

Sea $S_i = \{(\overrightarrow{o_{j,p}}, \psi_{j,v})/\overrightarrow{o_{j,p}} \in O_i, \psi_{j,v} = \Psi(\overrightarrow{o_{j,p}}, \overrightarrow{v})\}$, obtenido mediante kNN(G[i], $\overrightarrow{v}$,k) (Algoritmo 3).

$S^{'} = S^{'} \cup S_i$

end for

Sea $S^{''}$ la lista de los elementos de $S^{'}$ ordenada de mayor a menor según el valor de $\psi_{j,v}$ de cada elemento.

Sea $S = \{S^{''}[1], S^{''}[2], ..., S^{''}[k]\}$ las k soluciones más próximas al objeto de consulta.

return $\{r_j / r_j \in R, r_j = \tilde{\zeta}(o_{i,p}), (o_{i,p}, \psi_{i,v}) \in S\}$.

altamente dispersos, con un número suficientemente grande de estos la carga de trabajo de los procesadores tiende a equilibrarse "de manera natural".

D. Paralelización de las consultas

El proceso de buscar las soluciones parciales que aporta cada grafo al resultado de una consulta es completamente independiente para cada uno de ellos. Posteriormente es necesario combinar estas soluciones parciales para obtener la solución final, proceso que sigue siendo de naturaleza secuencial y es similar al descrito para la versión secuencial.

Los cambios con respecto al Algoritmo 2 son mostrados en el Algoritmo 4. Notese que el bucle `for` se ejecuta en paralelo por todos los procesadores involucrados. Se ha incluido además una sección crítica para evitar que más de un proceso actualice el conjunto $S^{'}$ en el mismo instante, lo que provocaría inconsistencias en el mismo.

IV. Análisis experimental

A. Entorno experimental

Para realizar nuestros experimentos usamos un repositorio de la agencia Reuters con los textos de un gran número de noticias cortas en prensa: la colección Reuters corpus versión 1 (RCV1-v2) [7].

El vector de características $\overrightarrow{o_i}$ de cada documento r_i fue producido mediante la concatenación de los textos en los elementos `<headline>` y `<text>` de los XML originales. El texto fue reducido a caracteres en minúsculas, después de lo cual se realizó sobre los mismos un proceso de tokenización, eliminación de puntuaciones y estemizado. Se eliminaron las palabras de parada, se sopesaron los términos y finalmente los pesos fueron normalizados, tras realizar una selección de características.

Algoritmo 3 Función kNN

Entrada: G_i, estructura de indexado; $\overrightarrow{v}$, el vector de consulta; k, la cantidad de soluciones requerida.
Salida: KNN, los k objetos más próximos a $\overrightarrow{v}$.

{**Determinando los vértices de entrada**}.
Sean $G_i = G[i]$, $O_i = G_i[1]$,$A_i = G_i[2]$ y $E_i = G_i[3]$.
Sea $S^{'} = \overrightarrow{o_{k_1,i}}, \overrightarrow{o_{k_2,i}}, ..., \overrightarrow{o_{k_{|S|},i}}$ la lista de elementos de E_i, ordenada en orden decreciente de su proximidad a $\overrightarrow{v}$.
Sea $\alpha \in \{1, ..., |S|\}$ una constante predefinida que determina del conjunto S el número real de vértices que serán usados como puntos de entrada a la estructura de indexado.
{**Búsqueda del vecino más próximo a** $\overrightarrow{v}$}.
Sea $NN = []$ una lista vacía.
for $e = 1 \rightarrow \alpha$ **do**
  $\overrightarrow{s_0} = S^{'}[e]$.
  repeat
    Sea L la lista ordenada de los elementos de $\{\overrightarrow{o_{j,i}}/\overrightarrow{o_{j,i}} \in G_i, (\overrightarrow{s_0}, \overrightarrow{s_{j,i}}) \in A_i\}$ en orden decreciente de su proximidad a $\overrightarrow{v}$.
    if $\Psi(L[1], \overrightarrow{v}) > \Psi(\overrightarrow{s_0}, \overrightarrow{v})$ **then**
      $\overrightarrow{s_0} = L[1]$.
    end if
  until $\Psi(L[1], \overrightarrow{v}) \leq \Psi(\overrightarrow{s_0}, \overrightarrow{v})$
  $NN[e] = \overrightarrow{s_0}$.
end for
Ordenar los elementos de NN en orden decreciente de su proximidad a $\overrightarrow{v}$.
$kNN = NN[1]$.
{**Búsqueda de los siguientes** $(k-1)$ **vecinos más próximos a** $\overrightarrow{v}$}
if $k > 1$ **then**
  $kNN = kNN \cup \{\overrightarrow{o_{j,i}}/\overrightarrow{o_{j,i}} \in G_i, (\overrightarrow{s_0}, \overrightarrow{o_{j,i}}) \in A_i\}$.
  repeat
    Sea L la lista ordenada de los elementos de kNN en orden decreciente de su proximidad a $\overrightarrow{v}$.
    Sea $L^{'} = L[1], L[2], ..., L[t]$ tal que $t = min(k-1, |L|)$.
    $kNN = \{\overrightarrow{o_{j,i}}/\overrightarrow{o_{j,i}} \in L^{'}\}$.
    Sea $\rho = \Psi(L[t], \overrightarrow{v})$ la mínima proximidad entre los elementos de $L^{'}$ y $\overrightarrow{v}$.
    for $p = 1 \rightarrow t$ **do**
      Sea $H_p = \{\overrightarrow{o_{j,i}}/\overrightarrow{o_{j,i}} \in G_i, (L^{'}[p], \overrightarrow{o_{j,i}}) \in A_i, \Psi(\overrightarrow{o_{j,i}}, \overrightarrow{v}) \geq \rho)\}$
      Sea $kNN^{'} = kNN \cup H_p$.
    end for
    $kNN = kNN^{'}$.
  until $(\left|kNN^{'}\right| \geq k) \wedge (kNN^{'} = kNN)$
end if
return kNN.

Algoritmo 4 BuscarKPróximosPar(R, G, q, k)

{...}
for $i = 1 \rightarrow N$ **do** {en paralelo}
  Sea $S_i = \{(\overrightarrow{o_{j,p}}, \psi_{j,v})/\overrightarrow{o_{j,p}} \in O_i, \psi_{j,v} = \Psi(\overrightarrow{o_{j,p}}, \overrightarrow{v})\}$, obtenido mediante kNN(G[i],$\overrightarrow{v}$,k) (Algoritmo 3).
  Sección crítica: $S^{'} = S^{'} \cup S_i$
end for
{...}

Se usó de la partición LYRL2004 su conjunto de entrenamiento que contiene 23.149 vectores. El espacio de representación de este conjunto tiene 47.152 dimensiones. La matriz de representación de este espacio contiene una gran número de ceros, debidos a la alta dispersión de los vectores.

Todos los experimentos fueron realizados sobre un multiprocesador SGI Altix 350 con memoria compartida. Esta máquina tiene 8 nodos, cada uno de los cuales tiene 2 procesadores Intel Itanium2, a 1.5 GHz. Cada nodo cuenta con 4Gb de memoria local conectadas mediante una red SGI NUMAlink, llegando a 32 GBytes de memoria compartida.

Todos los algoritmos fueron implementados usando lenguaje C, y fueron compilados con ICC 9.0 sobre el sistema operativo Linux. Las versiones paralelas fueron implementas usando la interfaz OpenMP.

B. Resultados experimentales

Para estudiar el comportamiento de nuestra propuesta se generó, para los esquemas secuencial y paralelo, un gran número de estructuras de indexado con diferentes números de grafos. Para seleccionar el valor de θ se generaron grafos usando valores entre 10 y 90. Finalmente seleccionamos el valor de 50 por ser el que garantizó un mejor equilibrio entre la calidad de los resultados y la selectividad del método.

Los resultados fueron obtenidos mediante la técnica de validación cruzada **10-fold-cross validation**, por lo que se procesaron 10 repositorios distintos. Los valores experimentales mostrados son los promedios de los obtenidos para cada repositorio.

Por cada experimento medimos el coste temporal de la generación de la estructura de indexado para 20.000 objetos, el coste temporal de 2.300 consultas, el consumo de memoria principal, la selectividad y la dimensionalidad de cada espacio de representación.

La asignación de objetos a los subconjuntos de datos fue realizada de manera aleatoria.

C. Precisión de los resultados

El primer primer aspecto estudiado fue la calidad de los resultados obtenidos durante las consultas. Los comparamos con los obtenidos con el método exhaustivo, que garantiza las soluciones exactas.

La Figura 1 muestra que la exactitud de los resultados se incrementa con el número de grafos usados. El incremento fue desde el 92% de soluciones exactas para un grafo hasta casi el 97% para 16.

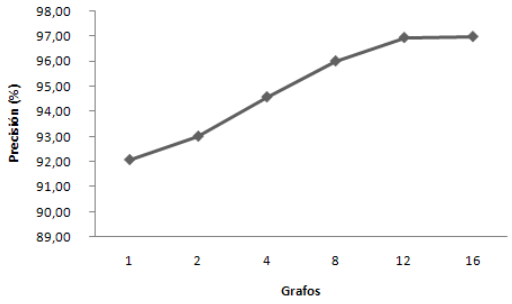

Fig. 1. Precisión de los resultados comparados contra los obtenidos mediante el método exhaustivo.

D. Selectividad

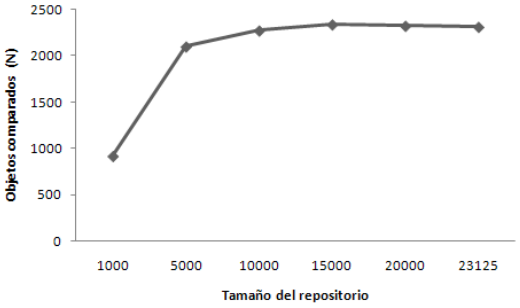

Fig. 2. Número de objetos evaluados como promedio durante las consultas, variando el tamaño del repositorio.

Como se aprecia en la Figura 2, el número promedio de objetos comparados durante las consultas se va estabilizando a medida que se incrementa el tamaño del repositorio, por lo que la selectividad del método crece en igual medida.

E. Dimensionalidad del espacio de representación

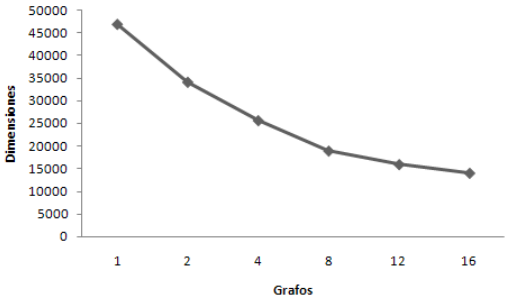

Fig. 3. Dimensionalidad máxima obtenida para los espacios de representación, variando el número de grafos usados.

Para analizar la dimensionalidad del espacio de representación de cada problema usamos el tamaño de los centroides de cada grafo, y tomamos en cada caso el mayor valor. Por ejemplo, cuando usamos un solo grafo tomamos el tamaño de su centroide, pero cuando generamos más de un grafo medimos el tamaño del centroide de cada grafo y mostramos el mayor valor.

La tendencia seguida por la dimensionalidad del espacio es mostrada en la Figura 3. Como se muestra, a medida que se usan más grafos para indexar el repositorio, el tamaño máximo de los espacios de representación disminuye. Esto se debe a que cuantos más grafos son usados, los subconjuntos de datos contienen menos elementos, y debido a la dispersión de los datos, van a contener en conjunto menos dimensiones. Esto es así porque aumenta la probabilidad de que en cada submatriz de representación, que contiene de la matriz original solamente aquellas filas de los vectores que le pertenecen, exista una gran cantidad de columnas con todos los valores a 0, en cuyo caso esa dimensión del espacio original no se tiene en cuenta.

La Figura 3 muestra que la máxima dimensionalidad del espacio de representación disminuye desde aproximadamente 50.000 dimensiones hasta casi 14.000, cuando se varía el número de grafos desde 1 hasta 16. Es necesario notar que el problema general mantiene siempre la misma dimensionalidad, pero las dimensiones son repartidas de manera aleatoria entre los grafos. Una misma dimensión puede aparecer en más de un grafo.

F. Coste temporal de la generación de los grafos

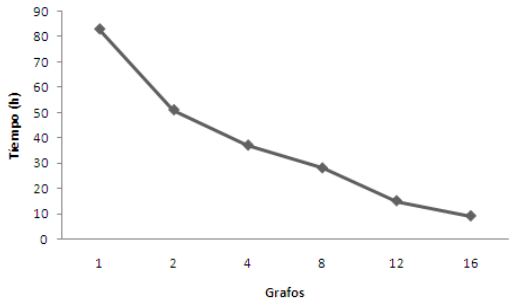

Fig. 4. Coste temporal para generar los grafos en la variante secuencial.

La Figura 4 muestra el coste temporal cuando los grafos son generados de manera secuencial, variando el número de grafos desde 1 hasta 16. Para la versión paralela usamos, de modo natural, la misma cantidad de procesadores que de grafos generados.

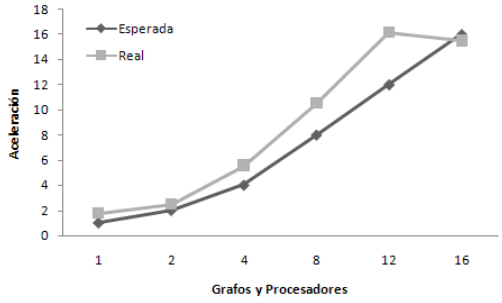

Fig. 5. Aceleración optenida con la versión paralela.

La Figura 5 muestra la aceleración optenida con la versión paralela de generación de los grafos. Esta muestra además que se han obtenido superaceleraciones en casi todos los casos. Esto se debe al mejor

uso de la estructura de la memoria cuando se usan varios procesadores que cuando se usa uno solo.

G. Consumo de memoria por los grafos

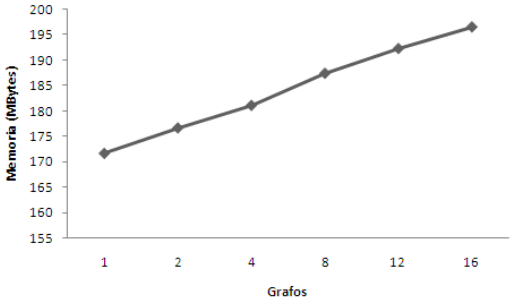

Fig. 6. Consumo de memoria principal, variando el número de grafos.

La Figura 6 muestra la cantidad de memoria necesaria para almacenar los grafos. Los valores son los mismos tanto para la versión secuencial como para la paralela.

Los resultados demuestran que el consumo de memoria se incrementa poco a medida que son usados más grafos para indexar el repositorio. Este incremento lento se debe a que cuantos más grafos son usados cada uno de estos contiene menos objetos, pero cada grafo aporta un consumo adicional debido a que al mantener la cantidad de vecinos de cada objeto igual a θ en cada grafo, el número global de enlaces se incrementa. Además, para cada grafo se almacena un conjunto de objetos de entrada.

H. Coste temporal de las búsquedas

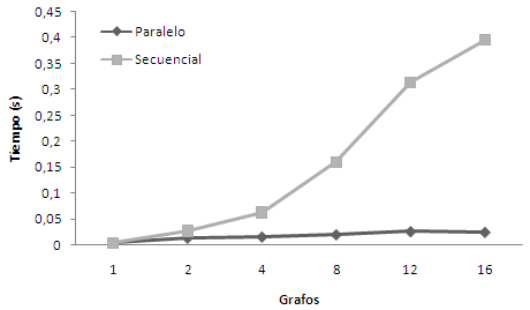

Fig. 7. Coste temporal para generar los grafos en la variante secuencial.

La Figura 7 muestra el promedio del coste temporal necesario para evaluar una consulta, variando el número de grafos, tanto para la versión secuencial como para la paralela.

Se puede observar que este coste se incrementa a medida que crece el número de grafos. Esto es debido a que se necesita, entre otros aspectos, evaluar un número mayor de vértices de entrada. También se observa que este incremento disminuye drásticamente en la versión paralela, debido a que sin importar el número de grafos usados, las consultas a cada grafo se solapan en el tiempo. En este caso, el incremento del coste temporal se debe fundamentalmente a la pequeña sección secuencial remanente, en la que se obtiene el resultado final a partir de los parciales obtenidos para cada grafo.

V. Conclusiones y trabajo futuro

Hemos presentado un método de acceso para espacios de muy alta dimensionalidad basado en el uso de multiples grafos como estructura de indexado. La estrategia de búsqueda que utiliza, aunque da lugar a soluciones aproximadas, lo hace con un nivel de error bajo y con una alta selectividad. Esto provoca bajos costes temporales durante las consultas.

Como hemos observado a medida que se usan más grafos en la estructura de indexado la precisión del método se incrementa, aunque el coste temporal de las consultas se incrementa un poco.

La estructura de indexado se ha elegido además, porque permite su construcción y utilización en paralelo de forma sencilla y eficiente. Los resultados experimentales demuestran que se obtienen muy buenas prestaciones con la versión paralela de ambos procesos.

Como trabajo futuro explotaremos las características de arquitecturas híbridas, con la combinación del uso de memoria compartida y distribuida.

Agradecimientos

El presente trabajo ha sido financiado por el Departamento Ingeniería y Ciencia de los Computadores de la Universidad Jaume I, Castellón.

Referencias

[1] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufman, ISBN 0123694469, 2006.

[2] Cortizo, J.C., Giráldez, J.I., *Multi criteria wrapper improvements to naive bayes learning*. LCNS, Vol. 4224, 2006.

[3] Berchtold, S. et al., *Independed quantization: an index compression technique for high-dimensional data spaces*. ICDE'00, pp. 577–588, 2000.

[4] Zhenjie Zhang and Beng Chin and Ooi Srinivasan and Parthasarathy Anthony and K. H. Tung, *Similarity search on bregman divergence: Towards non-metric indexing*, In VLDB, 2009.

[5] Ro J. S. Dutra and William A. Pearlman and Eduardo A. B. Da Silva and Senior Member, *Successive Approximation Wavelet Coding of 1 AVIRIS Hyperspectral Images*, 2010.

[6] Edgar Chavez, Karina Figueroa y Gonzalo Navarro, *Effective Proximity Retrieval by Ordering Permutations*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). Vol. 30 No. 9. pp 1647-1658, 2007.

[7] Lewis, D. D. et al., *RCV1: A new benchmark collection for text categorization research*, Journal of machine learning research, num. 5, pp. 361-397, 2004.