Optimización en Memoria Secundaria para el *EGNAT*

Roberto Uribe Paredes*
Depto. de Ingeniería en Computación
Universidad de Magallanes, Bulnes 01855, Punta Arenas, Chile.
Grupo de Bases de Datos - UART
Universidad Nacional de la Patagonia Austral, Río Turbio, Argentina.
(ruribe@ona.fi.umag.cl)

Mauricio Quintullanca Salgado**
Depto. de Ingeniería en Computación
Universidad de Magallanes, Bulnes 01855, Punta Arenas, Chile.
(mquintu@ona.fi.umag.cl)

Abstract

The *EGNAT* is a data structure deviced for searching in metric spaces. Empirical results show that this structure achieves good performance for high-dimensional metric spaces. This structure is tree type and is based on clustering, It has characteristic of to be dynamic and Secundary Memory enhanced.

This paper describe a different choice for storage on secundary memory for this structure, proving that is possible keep cost low of storage on disc and diminishing the cost for distance evaluations.

Keywords: databases, data estructures, algorithms, metric spaces, similarity queries.

Resumen

El *EGNAT* es una estructura de datos para búsquedas por similaridad en espacios métricos. Esta estructura ha demostrado tener buen desempeño en espacios de alta dimensión. Esta estructura es del tipo arreglo y basada en clustering, tiene la característica de ser dinámica y optimizada para memoria secundaria.

El presente trabajo describe una alternativa de almacenamiento en memoria secundaria para la estructura, demostrando que se puede disminuir los costos de almacenamiento y reducir los cálculos de distancia durante la búqueda.

Palabras claves: bases de datos, estructuras de datos, algoritmos, espacios métricos, consultas por similaridad.

1. Introducción

1.1. Antecedentes

Uno de los problemas de gran interés en ciencias de la computación es el de "búsqueda por similaridad", es decir, encontrar los elementos de un conjunto más similares a una muestra. Esta búsqueda es necesaria en

*Parcialmente financiado por Fondecyt 1060776, Conicyt, Chile.
**Parcialmente financiado por el programa de investigación PR-F1-002IC-06, Universidad de Magallanes.

múltiples aplicaciones, como ser en reconocimiento de voz e imagen, compresión de video, genética, minería de datos, recuperación de información, etc. En casi todas las aplicaciones la evaluación de la similaridad entre dos elementos es cara, por lo que usualmente se trata como medida del costo de la búsqueda la cantidad de similaridades que se evalúan.

Interesa el caso donde la similaridad describe un espacio métrico, es decir, está modelada por una función de distancia que respeta la desigualdad triangular. En este caso, el problema más común y difícil es en aquellos espacios de "alta dimensión" donde el histograma de distancias es concentrado, es decir, todos los objetos están más o menos a la misma distancia unos de otros.

El aumento de tamaño de las bases de datos y la aparición de nuevos tipos de datos sobre los cuales no interesa realizar búsquedas exactas, crean la necesidad de plantear nuevas estructuras para búsqueda por similaridad o búsqueda aproximada. Asimismo, se necesita que dichas estructuras sean dinámicas, es decir, que permitan agregar o eliminar elementos sin necesidad de crearlas nuevamente. Así también, las aplicaciones reales requieren que dichas estructuras permitan ser almacenadas en memoria secundaria eficientemente, como también que posean métodos optimizados para reducir los costos de accesos a disco.

1.2. Marco teórico

La similaridad se modeliza en muchos casos interesantes a través de un espacio métrico, y la búsqueda de objetos más similares a través de una búsqueda por rango o de vecinos más cercanos.

Definición 1 (*Espacios Métricos*): Un espacio métrico es un conjunto X con una función de distancia $d : X^2 \to R$, tal que $\forall x, y, z \in X$,

1. $d(x, y) \geq 0 \, and \, d(x, y) = 0 \, ssi \, x = y$. *(positividad)*
2. $d(x, y) = d(y, x)$. *(Simetría)*
3. $d(x, y) + d(y, z) \geq (d(x, z)$. *(Desigualdad Triangular)*

Definición 2 (*Consulta por Rango*): Sea un espacio métrico *(X,d)*, un conjunto de datos finito $Y \subseteq X$, una consulta $x \in X$, y un rango $r \in R$. La consulta de rango alrededor de x con rango r es el conjunto de puntos $y \in Y$, tal que $d(x, y) \leq r$.

Definición 3 (*Los k Vecinos más Cercanos*): Sea un espacio métrico *(X,d)*, un conjunto de datos finito $Y \subseteq X$, una consulta $x \in X$ y un entero k. Los k vecinos más cercanos a x son un subconjunto A de objetos de Y, donde la $|A| = k$ y no existe un objeto $y \in A$ tal que *d(y,x)* sea menor a la distancia de algún objeto de A a x.

El objetivo de los algoritmos de búsqueda es minimizar la cantidad de evaluaciones de distancia realizadas para resolver la consulta. Los métodos para buscar en espacios métricos se basan principalmente en dividir el espacio empleando la distancia a uno o más objetos seleccionados. El no trabajar con las características particulares de cada aplicación tiene la ventaja de ser más general, pues los algoritmos funcionan con cualquier tipo de objeto [4].

Existen distintas estructuras para buscar en espacios métricos, las cuales pueden ocupar funciones discretas o continuas de distancia. Algunos son BKTree [3], MetricTree [8], GNAT [2], VpTree [10], FQTree [1], MTree [5], SAT [6], Slim-Tree [7].

Algunas de las estructuras anteriores basan la búsqueda en pivotes y otras en clustering. En el primer caso se seleccionan pivotes del conjunto de datos y se precalculan las distancias entre los elementos y los pivotes. Cuando se realiza una consulta, se calcula la distancia de la consulta a los pivotes y se usa la desigualdad triangular para descartar candidatos.

Los algoritmos basados en clustering dividen el espacio en áreas, donde cada área tiene un *centro*. Se almacena alguna información sobre el área que permita descartar toda el área mediante sólo comparar la consulta con su centro. Los algoritmos de clustering son los mejores para espacios de alta dimensión, que es el problema más difícil en la práctica.

Existen dos criterios para delimitar las áreas en las estructuras basadas en clustering, *hiperplanos* y *radio cobertor* (*covering radius*). El primero divide el espacio en particiones de *Voronoi* y determina el hiperplano al cual pertenece la consulta según a qué centro corresponde. El criterio de radio cobertor divide el espacio en esferas que pueden intersectarse y una consulta puede pertenecer a más de una esfera.

Definición 4 (*Diagrama de Voronoi*): Considérese un conjunto de puntos $\{c_1, c_2, \ldots, c_n\}$(centros). Se define el diagrama de Voronoi como la subdivisión del plano en n áreas, una por cada c_i, tal que $q \in$ al área c_i sí y sólo sí la distancia euclidiana $d(q, c_i) < d(q, c_j)$ para cada c_j, con $j \neq i$.

Existen dos características poco comunes en las estructuras actuales. La primera es el *dinamismo*, es decir, la posibilidad de insertar y eliminar objetos una vez construida la estructura. La segunda característica, aún menos frecuente, es la manipulación de estas estructuras en memoria secundaria. No poseer dichas características hace poco factible la utilización de estas estructuras en aplicaciones reales.

El ***egnat*** es una estructura basada principalmente en el diagrama de Voronoi, aunque igualmente usa radio cobertor. Está basada en el *gnat* [2] que es una generalización del *Generalized Hyperplane Tree* (*GHT*) [8].

El presente trabajo propone alternativas para reducir el tamaño en disco de la estructura, manteniendo los costos en términos de evaluaciones de distancia.

Para este artículo se seleccionaron las pruebas realizadas sobre dos espacios métricos. El primero, un diccionario de palabras en castellano de 86,061 objetos, donde la distancia utilizada es la *distancia de edición*, la cual entrega como resultado el número mínimo de inserciones, eliminaciones o reemplazos de caracteres para que una palabra sea igual a otra. El segundo es un espacio de vectores de coordenadas reales de dimensión 10 generados con distribución de *Gauss* con media 1 y varianza 0.1 cuya cantidad de objetos es de 100,000, para este espacio se utilizó la *distancia Euclidiana*. Para la búsqueda se creó la estructura con el 90 % de los datos y se reservo el 10 % como consultas. Se considera que estos espacios muestran claramente el comportamiento del *egnat*.

2. *gnat Evolutivo (Evolutionary gnat)*

El *gnat evolutivo* o *egnat* presentado en [9] es una estructura de datos completamente dinámica, competitiva para búsquedas por similaridad en espacios métricos de alta dimensión y con buen desempeño en memoria secundaria. El *egnat* pertenece al grupo de algoritmos basados en particiones compactas y es una optimización en memoria secundaria para el *gnat* en términos de espacio, accesos a disco y evaluaciones de distancia.

El *egnat* es un árbol que posee dos tipos de nodos, un *nodo bucket* y un *nodo gnat*. Los nodos nacen como bolsas o buckets y cuya única información es sólo la distancia al padre, al estilo de las estructuras basadas en pivotes (un solo pivote). Este mecanismo es el que permite disminuir el espacio requerido en disco para almacenar la estructura y logra mantener un buen desempeño en términos de evaluaciones de distancia. Si un nodo se completa, éste evoluciona de un *nodo bolsa* a un *nodo gnat*, reinsertando los objetos de la bolsa al nuevo *nodo gnat* (ver figura 1).

2.1. Construcción

La construcción básica del nodo de tipo *gnat* es como sigue:

1. Se seleccionan k puntos (*splits*), $p_1, \ldots, p_k$ de la base de datos la cual se va a indexar.

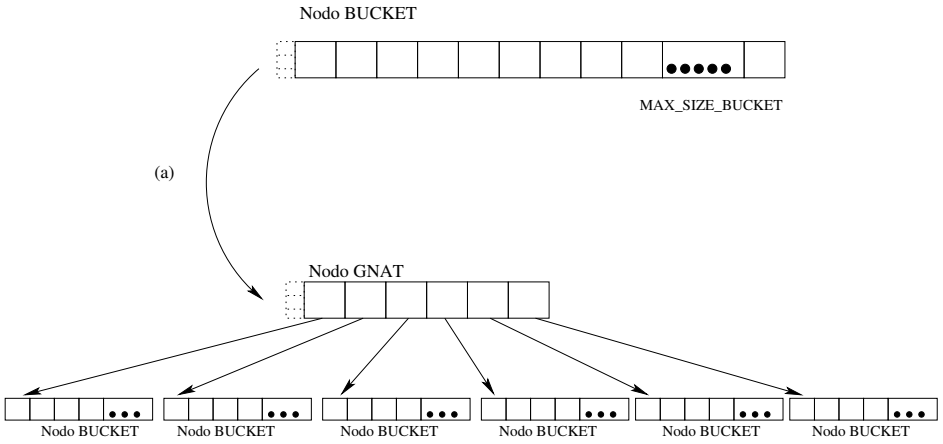

Figura 1: Mutación de un nodo bucket a un nodo gnat.

2. Se asocia cada punto restante del conjunto de datos al split más cercano a él. El conjunto de puntos asociados al split p_i se denota como D_{p_i}.

3. Para cada par de split (p_i, p_j), se calcula el rango $range\left(p_i, D_{p_j}\right) = \left[min\_d\left(p_i, D_{p_j}\right), max\_d\left(p_i, D_{p_j}\right)\right]$, una mínima y máxima distancia $Dist\left(p_i, x\right)$ donde $x \in D_{p_j} \cup \{p_j\}$.

4. El árbol se construye recursivamente para cada elemento en D_{p_i}.

Cada conjunto D_{p_i} va a representar un subárbol cuya raíz es p_i, o lo que es lo mismo, cada D_{p_i} va a corresponder al plano de Voronoi cuyo centro es p_i. En la figura 2 se muestra un ejemplo de construcción del primer nivel de un gnat con k=4, también se muestra la tabla de rangos que debe ser almacenada para cada centro p_i. En este ejemplo se insertaron los puntos en orden al valor numérico que tienen.

2.2. Búsqueda en el *egnat*

En una búsqueda se asume que se desean encontrar todos los puntos con distancia $d \leq r$ a el punto q. Durante la búsqueda en un *egnat* de determina en primer lugar si el nodo es de tipo *bucket* o *gnat*.

En el caso de que la búsqueda se realizara sobre un nodo de tipo bolsa, al mantener la distancia al centro se puede usar la desigualdad triangular para evitar el cálculo directo de la distancia de un objeto consulta sobre los objetos ubicados en los buckets de la siguiente manera:

- Sea q el objeto consulta, p el split que posee un hijo bucket, s_i los elementos ubicados dentro del bucket, r el radio de búsqueda, entonces, si

$$Dist(s_i,p) > Dist(q,p) + r \qquad o$$
$$Dist(s_i,p) < Dist(q,p) - r$$

  se puede determinar que el objeto s_i no está dentro del rango de búsqueda, de lo contrario hay que necesariamente hacer la comparación directa.

Esto se utiliza también para la búsqueda por rango 0 en las bolsas, lo que reduce considerablemente la cantidad de evaluaciones al momento de eliminar objetos.

Si el nodo es de tipo *gnat*, la búsqueda se realiza como fue definido originalmente en la estructura *gnat*.

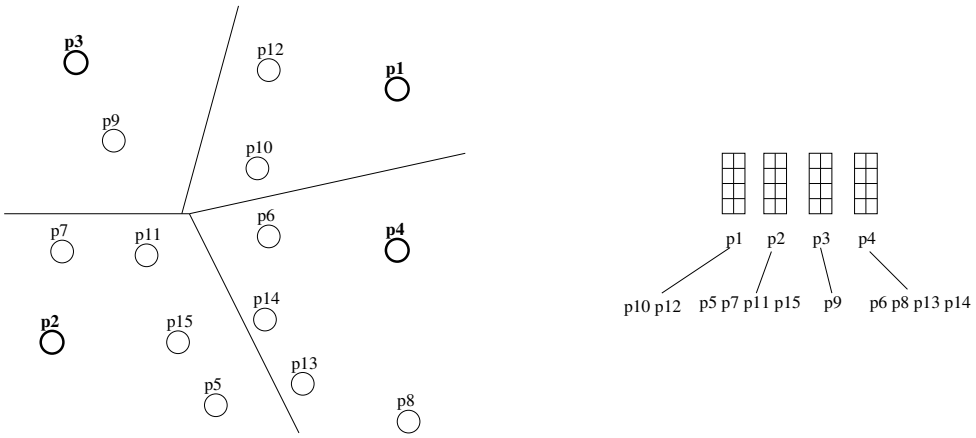

Figura 2: *egnat*: partición del espacio, representación de subplanos en un nodo de tipo *gnat*.

- Sea P la representación del conjunto de puntos splits del nodo actual el cual posiblemente contiene un vecino cercano a q. Inicialmente P contiene todos los puntos split del nodo actual. La búsqueda se realiza como se muestra en el algoritmo 1.

3. Alternativas de Almacenamiento

Uno de los nuevos parámetros a considerar en la construcción de estructuras diseñadas para memoria secundaria, es el espacio en disco utilizado. En este sentido, en el *egnat*, cada nodo, ya sea del tipo *gnat* o *bucket* equivale a una página de disco. Cada nodo del tipo gnat está completamente lleno, salvo que hubiesen eliminaciones, sin embargo, es muy frecuente que los nodos bucket tengan bajos porcentajes de utilización del nodo. Lo anterior provoca un aumento en la cantidad de nodos o páginas de disco y por ende un aumento en el tamaño de la estructura.

Uno de los problemas del egnat es el tamaño que requiere para almacenar los datos. Si bien logró reducir es espacio versus la estructura original gnat, aun requiere mucho más que el M-Tree, siendo ésta la ñunica desventaja del egnat frente al M-Tree.

La propuesta que se analiza en este artículo es poder compartir las páginas de disco por más de un nodo. En este sentido, a mayor cantidad de centros en un nodo gnat, su comportamiento es mucho mejor. Considerando además, que los nodos gnat están generalmente llenos y es en los bucket donde se tiene un bajo uso del nodo, entonces la propuesta es que una página de disco sea usada por más de un nodo bucket.

La figura 3 muestra la forma del árbol para la alternativa propuesta.

4. Resultados experimentales

Resultados experimentales se muestran en las tablas 2 y 3.

Esta idea de compatir la página por bucket o bolsas es basada principalmente en el hecho de que los promedios de uso de un nodo bolsa no son superiores a 14,52 para el caso del espacio de Gauss y de 17,06 para el espacio de

Algoritmo 1 *egnat*: búsqueda por rango r para la consulta q en un nodo de tipo *gnat*.

busquedarango(Nodo P, Consulta q, Rango r)

1: {Sea R el conjunto resultado}
2: $R \leftarrow \emptyset$
3: $d \leftarrow dist(p_0, q)$
4: **if** $d <= r$ **then**
5: se reporta p_0
6: **end if**
7: $range(p_0, q) \leftarrow [d - r, d + r]$
8: **for all** $x \in P$ **do**
9: **if** $range(p_0, q) \cap range(p_0, D_{p_x}) \neq \emptyset$ **then**
10: se agrega x a R
11: **if** $dist(x, q) <= r$ **then**
12: se reporta x
13: **end if**
14: **end if**
15: **end for**
16: **for all** $p_i \in R$ **do**
17: busquedarango(D_{p_i},q,r)
18: **end for**

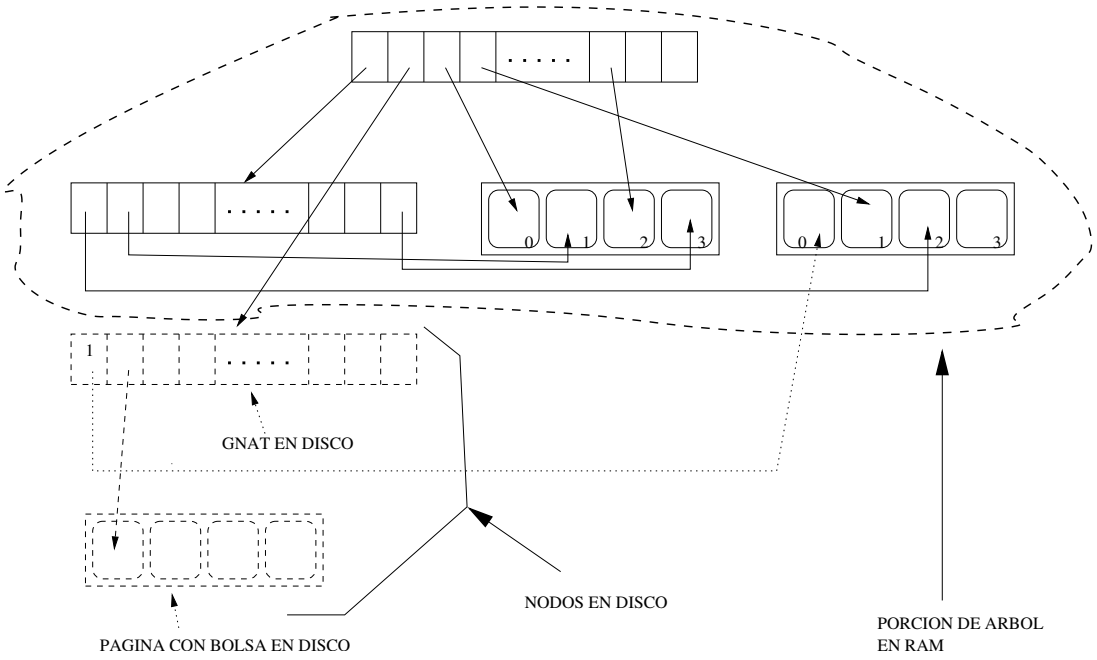

Figura 3: *egnat* : páginas compartidas por 4 buckets.

	Vectores de Gauss de dim. 10				Diccionario Español			
	centros	nodos x pág.	nro. pág.	b x obj	centros	nodos x pág.	nro. pág.	b x obj
gnat (4)	4	12	3.339	136.77	4	12	2.633	125,32
gnat (6)	6	7	5.722	234,37	6	7	3.648	173,62
gnat (8)	8	4	10.010	410,01	8	4	6.384	303,84
M-tree 0.4	-	-	4.461	182,72	-	-	1.585	75,44
M-tree 0.1	-	-	5.095	208,69	-	-	1.856	88,33
egnat	19	1	6.773	277,42	20	1	4.996	237,78
egnat B	19	4 (bolsas)	5.936	243,14	20	4 (bolsas)	4.157	197,85
egnat B	19	6 (bolsas)	4.822	197,51	20	6 (bolsas)	3.789	180,33

Tabla 2: Construcción: valores comparativos para páginas compartidas entre *gnat*, *M-tree, egnat* y *egnat* (versión final) con el 100 % de los datos.

	vectores de Gauss		diccionario en Español	
Buckets por pagina	4	6	4	6
MAX_SIZE_BUCKET	23	15	31	21
Cantidad de buckets	17.518	17.676	12.475	13.852
cantidad de gnat	1.556	1876	1.038	1.480
cantidad total de páginas	5.936	4.822	4.157	3.789
promedio de uso bucket	4,02 (17,48 %)	3,64 (24,27 %)	5,23 (16,87 %)	4,08 (19,43 %)

Tabla 3: *egnat* (versión final B): detalles de construcción para páginas con capacidad para 4 y 6 bolsas (100 % de los datos).

palabras. Por lo tanto las páginas podrían contener por ejemplo 4 bolsas, de esta manera la estructura aplicada a vectores tendría bucket de 23 centros y la estructura aplicada a palabras 31 centros.

En la tabla 2 se muestra la información para el egnat y el egnat con bolsa compartiendo páginas (versión B). Para este último experimento, la cantidad de páginas para el egnat es la suma de los nodos gnat más las páginas que contienen nodos de tipo bucket, en este caso 4 y 6 buckets por página. La misma tabla también muestra la columna promedio de byte por objeto (b x obj). En la misma tabla se muestra información adicional de una versión del gnat donde se comparte la página con varios nodos gnat y del M-Tree. Para el M-Tree, los valores 0.1 y 0.4 corresponden ael porcentaje mínimo utilizado en un nodo, garantizado por la estructura.

Para la alternativa mencionada para el egnat la información obtenida se muestra en la tabla 3, de ésta se desprende que al ser más pequeñas las bolsas, éstas se llenan más rápido, lo que implicó un aumento de los nodos de tipo gnat y de los bucket, pero ahora estos últimos utilizarían un cuarto o un sexto de la página.

De la tabla 2 se desprende que el M-Tree puede utilizar mejor el espacio dentro de una página en el espacio de palabras, siendo menos competitivo en el espacio de Gauss. Finalmente se puede indicar que la estructura egnat y el egnat con páginas compartidas por varios nodos logran reducir el espacio utilizado en disco y acercarse o mejorar los valores del M-tree.

Desde el punto de vista de las Evaluaciones de distancia, la versión B del egnat logra una leve mejora al reducir los cálculos de distancia (ver figura 4). En lo referente al tamaño en disco, la figura 5 muestra como la nueva versión igualmente mejora el espacio utilizado en disco, disminuyendo la cantidad de páginas requeridas por la estructura para almacenar la misma información.

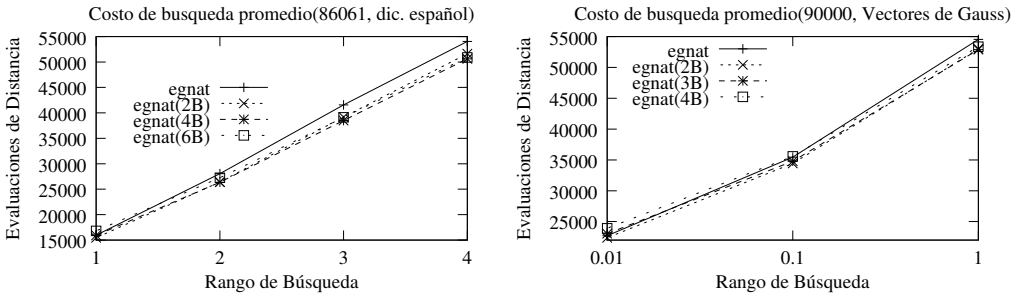

Figura 4: *egnat*: reducción de cálculos de distancia utilizando la versión B.

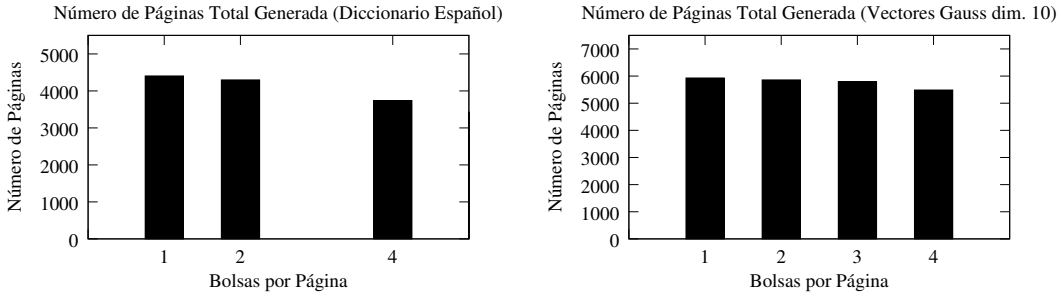

Figura 5: *egnat*: reducción de páginas de disco utilizando la versión B.

5. Conclusiones

5.1. Aspectos Relevantes y Aportes

Un gran número de las estructuras que han sido desarrolladas para búsquedas por similaridad están diseñadas sólo como prototipos para memoria principal. Por lo tanto, es relevante contar con estructuras sobre memoria secundaria de tal manera que puedan ser implementadas en aplicaciones reales. Considerando lo prometedor de la nueva estructura denominada *egnat*, desde el punto de vista de los costos en evaluaciones de distancia, se considera que el aporte más relevante del presente trabajo es la reducción del espacio utilizado en disco y de la reducción en los cálculos de distancia.

La nueva vesión también es competitiva con el M-Tree, en algunos casos mejorando el espacio requerido por la estructura.

Referencias

[1] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixedqueries trees. In *5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.

[2] Sergei Brin. Near neighbor search in large metric spaces. In *the 21st VLDB Conference*, pages 574–584. Morgan Kaufmann Publishers, 1995.

[3] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Communication of ACM*, 16(4):230–236, 1973.

[4] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José L. Marroquín. Searching in metric spaces. In *ACM Computing Surveys*, pages 33(3):273–321, September 2001.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree : An efficient access method for similarity search in metric spaces. In *the 23st International Conference on VLDB*, pages 426–435, 1997.

[6] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, 11(1):28–46, 2002.

[7] Caetano Traina, Agma Traina, Bernhard Seeger, and Christos Faloutsos. Slim-trees: High performance metric trees minimizing overlap between nodes. In *VII International Conference on Extending Database Technology*, pages 51–61, 2000.

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

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

[10] P. Yianilos. Data structures and algoritms for nearest neighbor search in general metric spaces. In *4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, 1993.