Parametrización local de espacios métricos *

Norma Edith Herrera
Departamento de Informática
Universidad Nacional de San Luis
Argentina
nherrera@unsl.edu.ar

Edgar Chávez
Escuela de Ciencias Físico - Matemáticas
Universidad Michoacana
México
elchavez@fismat.umich.mx

Resumen

Muchas aplicaciones en computación tienen por objetivo buscar objetos en una base de datos que sean similares a uno dado. Todas estas aplicaciones pueden tratarse en abstracto con el formalismo de ***espacio métrico***. Este método encapsula las propiedades de los objetos de la base de datos y permite construir índices genéricos.

Existen muchas técnicas de construcción de índices para realizar búsquedas de proximidad, todas las técnicas tienen parámetros que dependen de la geometría del espacio. Estos parámetros balancean el tiempo de construcción, el tiempo de búsqueda y la memoria utilizada por el índice.

En este trabajo presentamos un método de parametrización local que permite segmentar la base de datos de tal manera que a cada segmento se le pueden seleccionar de manera óptima sus parámetros adecuados.

Ilustramos la técnica probando con un índice particularmente difícil de parametrizar, el GNAT. Para este efecto seleccionamos el espacio métrico de cadenas de palabras bajo la distancia de edicion. La base de datos se divide en dos segmentos, los cuales se indizan por separado. Para satisfacer una consulta se busca en *ambos* índices. Esta operación resulta mas eficiente que buscar en el índice original.

Palabras claves: bases de datos, búsqueda por proximidad, espacio métrico.

*Este trabajo ha sido parcialmente subvencionado por CYTED VII.19 RIBIDI Project (ambos autores) y por CONACyT 36911-A (segundo autor)

1 Introducción

La búsqueda es un problema fundamental en Ciencias de la Computación, presente virtualmente en cada aplicación. Las bases de datos tradicionales se construyen basándose en el concepto de búsqueda exacta: la base de datos es dividida en registros, teniendo cada registro campos totalmente comparables; una consulta a la base retorna todos aquellos registros cuyos campos coincidan con los aportados en la búsqueda.

Actualmente las bases de datos han incluido la capacidad de almacenar nuevos tipos de datos tales como imágenes, sonido, video, etc. Estructurar este tipo de datos en registros, para adecuarlos al concepto tradicional de búsqueda exacta, es dificil en muchos casos y hasta imposible si la base de datos cambia mas rápido de lo que se puede estructurar (e.g. la `web`). Aún cuando pudiera hacerse, las consultas que se pueden satisfacer con la tecnología tradicional son limitadas a variaciones de búsquedas exactas. Nos interesan las búsquedas en donde se puedan recuperar objetos *similares* a uno dado. Este tipo de búsqueda se conoce con el nombre de ***búsqueda aproximada o búsqueda por similitud***, y surge en diversas áreas; reconocimiento de voz, reconocimiento de imágenes, compresión de texto, biología computacional, por nombrar algunas de ellas.

Todas estas aplicaciones tienen algunas características comunes. Existe un universo $\mathcal{X}$ de objetos y una función de distancia $d : \mathcal{X} \times \mathcal{X} \to R^+$ que modela la similitud entre los objetos. Esta función d cumple con las propiedades características de una función de distancia:

1. $d(x, y) = 0 \Leftrightarrow x = y$ (positividad estricta)
2. $d(x, y) = d(y, x)$(simetría)
3. $d(x, y) \leq d(x, y) + d(y, z)$ (desigualdad tringular)

El par $(\mathcal{X}, d)$ es llamado ***espacio métrico***. La base de datos es un conjunto $\mathcal{U} \subseteq \mathcal{X}$, el cual se preprocesa a fin de resolver búsquedas por similitud eficientemente. La medida de eficiencia está dada por el número de cálculos de distancia utilizados para resolver la búsqueda.

Básicamente, existen tres tipos de búsquedas de interés en espacios métricos, los cuales describimos con la notación estándar de la literatura [7]:

Búsqueda por rango : esta búsqueda, que denotaremos con $(q, r)_d$, consiste en recuperar todos los elementos de $\mathcal{U}$ que están a distancia r de un elemento q dado. En símbolos, $(q, r)_d = \{u \in \mathcal{U} : d(q, u) \leq r\}$

Búsqueda del vecino más cercano: esta búsqueda, que denotaremos con $nn(q)_d$, consiste en recuperar el (o los) elemento(s) más cercano(s) a un elemento q dado. En símbolos, $nn(q)_d = \{u \in \mathcal{U} : \forall v \in \mathcal{U} d(q, u) \leq d(q, v)\}$

Búsqueda de los k-vecinos más cercano la denotaremos con $nn_k(q)$; en este caso el objetivo es recuperar los k elementos más cercanos a q en $\mathcal{U}$. Esto significa encontrar un conjunto $A \subseteq \mathcal{U}$ tal que $|A| = k \wedge \forall u \in A, v \in \mathcal{U} - A : d(q, u) \leq d(q, v)$

Nuestro interés se ha centrado en las búsquedas por rango, dado que son la base para resolver los otros dos tipos de búsquedas .

Un caso particular de espacios métricos son los espacios vectoriales K-dimensionales, con las funciones de distancia $L_p((x_1 \ldots x_k), (y_1 \ldots y_k)) = (\sum_{1 \leq i \leq K} |x_i - y_i|^p)^{1/p}, p = 1 \ldots \infty$. Para estos

casos existen soluciones bien conocidas tales como KD-tree [2, 3], R-tree [10], etc. Sin embargo, éste no es el caso general.

Las investigaciones en la actualidad tienden al estudio de algoritmos en espacios métricos generales, donde existen varias técnicas conocidas para resolver el problema en un número sublineal de cálculos de distancia, con la condición del preprocesamiento de $\mathcal{U}$ [7]. Todas estas técnicas tienen parámetros que dependen de la geometría del espacio.

En este trabajo presentamos un método que permite particionar el espacio en dos segmentos, cada uno de ellos con ciertas características, de manera tal que se pueda realizar una selección de parámetros óptima a cada segmento.

Para probar esta técnica, hemos usado como índice una estructura particularmente difícil de parametrizar, el ***Geometric Near-neighbor Access Tree (GNAT)***. En particular hemos trabajado con las búsquedas por rango aplicadas a diccionarios de palabras, usando como función de distancia la distancia de edición (también llamada distancia de Leveshtein).

Comenzaremos dando el marco teórico del trabajo, para luego presentar la técnica de segmentación y los resultados experimentales en donde mostramos que, partir el diccionario en dos segmentos, indexar cada segmento separadamente y buscar también por separado es más eficiente que buscar en el espacio sin segmentar.

2 Geometric Near-neighbor Access Tree

Esta estructura, presentada por Sergey Brin en el año 1995 [4], es una extensión del *Generalized Hyperplane Tree (GHT)* [11]. El objetivo que se persigue, es que la estructura actúe como un modelo geométricamente jerárquico de los datos. Más específicamente, a partir del nodo raíz se obtiene una una idea de los datos como espacio métrico, y a medida que avanzamos en la jerarquía del árbol se logra una idea más exacta de la geometría de los mismos. Para lograr esto se construye una jerarquía basada en *Diagramas de Voronoi* [1].

La construcción de un GNAT de aridad m procede de la siguiente manera: en el primer nivel se seleccionan m pivotes $p_1, p_2, \cdots, p_m$ de $\mathcal{U}$, que se almacenan en el nodo raíz. A cada pivote p_i se le asocia el conjunto $\mathcal{U}_{p_i}$ formado por aquellos objetos que están más cerca de p_i que de cualquier otro pivote p_j; en símbolos:

$$\mathcal{U}_{p_i} = \{x \in \mathcal{U} / d(p_i, x) < d(p_j, x),\ \forall j = 1 \ldots m,\ j \neq i\}$$

Para cada $\mathcal{U}_{p_i}$, si su cardinalidad es mayor que m se construye recursivamente un GNAT, caso contrario se construye con esos elementos un nodo terminal.

En cada nodo del GNAT se almacena además una tabla, a la que llamaremos $rango$, de tamaño $O(m^2)$. Esta tabla mantiene información sobre las distancias mínimas y máximas, desde el pivote p_i a los conjuntos $\mathcal{U}_{p_j}$:

$$rango_{i,j} = [min_{x \in \mathcal{U}_{p_j}} d(p_i, x), max_{x \in \mathcal{U}_{p_j}} d(p_i, x)] \text{ , con } i, j = 1, \ldots, m$$

Esta información, junto con la desigualdad triangular, se usa en el momento de la búsqueda para descartar subárboles. Para una búsqueda por rango $(q, r)_d$, se compara q con algún pivote p_i, y se descartan todos aquellos pivotes p_j (y sus correspondientes $\mathcal{U}_{p_j}$) tales que

$$[d(q, p_i) - r, d(q, p_i) + r] \cap rango_{i,j} = \emptyset$$

La razón por la que esto puede hacerse es muy sencilla. Sea $y \in \mathcal{U}_{p_j}$, si $d(p_i, y) < d(p_i, q) - r$, luego por la desigualdad triangular se sigue que $d(p_i, y) < d(p_i, y) + d(y, q) - r$, de donde se deduce que $d(y, q) > r$. Análogamente, si $d(p_i, y) > d(p_i, q) + r$, por la desigualdad triangular se sigue que $d(p_i, q) + d(q, y) > d(p_i, q) + r$, de donde deducimos que $d(y, q) > r$ (ver figura 1).

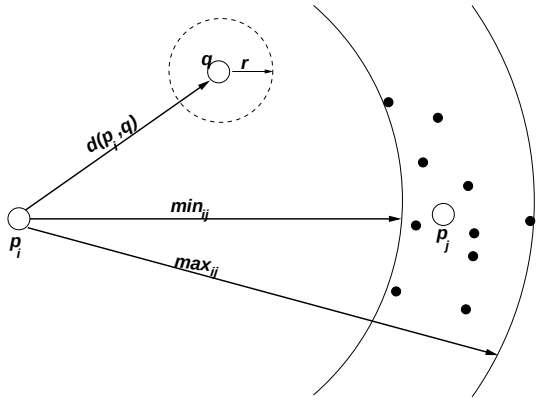

Figura 1: *Eliminación de subárboles usando* $rango_{ij}$. *En este caso podemos eliminar* $\mathcal{U}_{p_j}$ *dado que* $d(q, p) + r < min_{x \in \mathcal{U}_{p_j}} d(p_i, x)$

Este proceso se repite hasta que ningún pivote pueda descartarse. La búsqueda continúa luego recursivamente en aquellos subárboles no eliminados. Durante este proceso se agregan al resultado todos aquellos pivotes p_i tales que $d(p_i, q) \leq r$.

La *aridad* del árbol es el número de pivotes que se eligen en cada nodo. La aridad elegida influye notablemente en la performance del GNAT. Para algunos espacios métricos, una aridad alta puede ser una buena elección, mientras que para un espacio métrico diferente una aridad pequeña puede producir mejores resultados.

3 Dimensión Intrínseca

Las técnicas de indexación tradicionales para espacios vectoriales tienen una complejidad que depende exponencialmente de la dimensión del espacio. Las técnicas más recientes, han podido solucionar este problema y su complejidad sólo depende de la dimesión intrínseca del espacio; por ejemplo, un plano inmerso en un espacio vectorial de dimensión 50, tiene dimensión intrínseca 2. Este concepto de dimensión puede traducirse a espacios métricos generales y, como veremos, también afecta la performance de las técnicas de indexación.

Consideremos el histograma de distancias entre puntos en un espacio métrico $(\mathcal{X}, d)$. Este histograma se menciona en varios artículos como una medida fundamental relacionada a la dimensión intrínseca del espacio [4, 5, 6, 8, 9]. La idea es que, a medida que la dimensión intrínseca de un espacio métrico aumenta, la media μ del histograma crece y su varianza σ^2 se reduce.

La figura 2 da una idea intuitiva de por qué el problema de búsqueda se torna más difícil cuando el histograma es más concentrado. Consideremos una búsqueda $(q, r)_d$, y un índice basado en pivotes elegidos aleatoriamente. Los histogramas de la figura representan posibles distribuciones de distancias entre q y algún pivote p. La regla de eliminación dice que podemos descartar aquellos puntos y tales que $y \notin [d(p, q) - r, d(p, q) + r]$. Las áreas sombreadas muestran los puntos que no podrán

descartarse. A medida que el histograma se concentra más alrededor de su media, disminuye la cantidad de puntos que pueden descartarse usando como dato $d(p, q)$. Este fenómeno es independiente de la naturaleza del espacio métrico, y nos brinda una forma de cuantificar cuán dura es una búsqueda sobre el mismo.

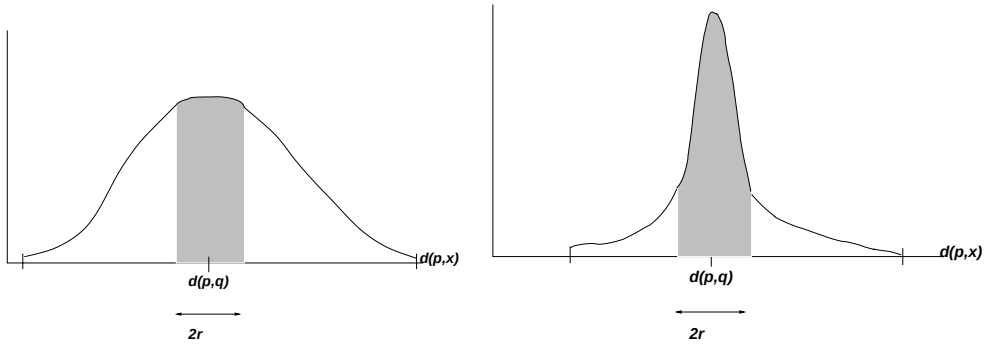

Figura 2: *Histogramas de distancias de baja dimensionalidad (izquierda), y de alta dimensionalidad (derecha)*

Este concepto se formaliza a través de la siguiente definición :

Definición: la dimensión intrínseca de un espacio métrico $(\mathcal{X}, d)$ se define como $\rho = \frac{\mu}{\sigma^2}$, siendo μ y σ^2 la media y la varianza respectivamente de su histograma de distancia.

Bajo esta definición, una base de datos obtenida como una muestra uniformemente distribuida de un espacio vectorial de dimensión k, tiene dimensión intrínseca $\Theta(k)$. Esto significa que la definición dada concuerda con el concepto de dimensión en espacios vectoriales.

En forma similar a lo que sucede en espacios vectoriales, todos los algoritmos degradan sistemáticamente cuando ρ se incrementa [7].

4 Estrategia de Partición

Descubrir la estructura subyacente del conjunto de datos es sumamente útil en el diseño de algoritmos de indexación. En particular, saber cómo se agrupan los elementos del espacio métrico nos servirá para identificar la zona de búsqueda más difícil.

Una forma de visualizar la distribución de los datos del espacio métrico, es por medio de los histogramas de distancias. Dado un espacio métrico $(\mathcal{X}, d)$ y un elemento $p \in \mathcal{X}$, *el histograma local* respecto del punto de referencia p es la distribución de distancias de p a los elementos $x \in \mathcal{X}$.

El histograma local puede ser muy diferente del histograma global; pero si los histogramas locales de diferentes puntos de referencia son similares, entonces podemos predecir a través de ellos la distribución de los elementos de una base de datos $\mathcal{U} \subseteq \mathcal{X}$.

Si un grupo de elementos se encuentra simultáneamente en la zona central de los histogramas de varios puntos de referencia, entonces ellos conformarán una zona de búsqueda difícil de tratar, dado que están en la zona de mayor concentración de elementos. Llamaremos a este grupo de elementos ***núcleo duro*** del espacio y lo denotaremos con $nd(\mathcal{X}, d)$. Los restantes elementos formarán lo que llamaremos el ***núcleo blando***, que denotaremos con $nb(\mathcal{X}, d)$,

Luego, dada una base de datos $\mathcal{U} \subseteq \mathcal{X}$ identificar y eliminar su núcleo duro indexándolo separadamente, permitirá mejorar la performance dado que las búsquedas sobre los restantes elementos no se verán afectadas por ese conglomerado de elementos. Esto implica:

- Particionar la base de datos $\mathcal{U}$ en $nd(\mathcal{U}, d)$ y $nb(\mathcal{U}, d)$.
- Indexar separadamente $nd(\mathcal{U}, d)$ y $nb(\mathcal{U}, d)$
- Resolver las búsquedas $(q, r)_d$ sobre $\mathcal{U}$, buscando en $nd(\mathcal{U}, d)$ y $nb(\mathcal{U}, d)$

El método de detección de $nd(\mathcal{U}, d)$ es sencillo, sólo debemos hacer intersecciones de las zonas centrales de histogramas locales para distintos puntos de referencia p. Habiendo obtenido $nd(\mathcal{U}, d)$, la detección de $nb(\mathcal{U}, d)$ es trivial.

La figura 3 describe el proceso de detección del núcleo duro de una base de datos $\mathcal{U}$. El parámetro s es el porcentaje de elementos que deben quedar finalmente en $nd(\mathcal{U}, d)$. El parámetro rc es el radio de corte utilizado para detectar la zona de mayor conflicto del histograma local del punto p (denotado en el código como $hl(p)$). La idea es, teniendo calculado $hl(p)$, tomar los elementos que se encuentren alrededor de la mediana del histograma.

```
Obtener_nd(in U, in s, in rc, out nd(U,d))

1. nd(U,d) ← U
2. Elegir un punto p ∈ U
3. Mientras |nd(U,d)| > s · |U| hacer
4.     Calcular hl(p)
5.     m ← mediana(hl(p))
6.     nd(U,d) ← nd(U,d) ∩ {x ∈ U / d(x,p) ∈ [m − rc, m + rc]}
7.     Elegir un punto p ∈ U − nd(U,d)
8. Fin mientras
```

Figura 3: *Algoritmo de detección de* $nd(\mathcal{U}, d)$

5 Análisis experimental

ESPACIO MÉTRICO

Los experimentos fueron realizados sobre diccionarios de palabras usando como función de distancia la distancia de edición. Esta función es discreta y calcula la mínima cantidad de palabras que hay que agregar, cambiar y/o eliminar a una palabra para obtener otra. Este modelo es comúnmente usado en recuperación de texto, procesamiento de señales y aplicaciones de biología computacional.

PARÁMETROS USADOS EN LA CONSTRUCCIÓN DEL GNAT

Se trabajó con política de selección de pivotes aleatoria. La cantidad de pivotes utilizada en cada caso, dependió de la de la base de datos a indexar. Hemos podido comprobar experimentalmente que, si el tamaño de la base de datos aumenta, aumentar la aridad del GNAT no siempre es una buena decisión.

Los experimentos persiguieron entonces dos objetivos. Primero analizar el comportamiento de la técnica de particionamiento, haciendo uso de una cantidad de memoria no superior a la que la que se utilizaría si no se particionara el espacio. Para ello se realizaron experimentos de manera tal que en los núcleos se usara una aridad menor o igual a la utilizada para indexar el diccionario sin particionar.

Luego, se analizó el comportamiento del método de particionamiento, usando las aridades más convenientes en cada caso, aún cuando esto implicara que en los núcleos se utilizara una cantidad de memoria superior a la necesitada para indexar el diccionario sin particionar.

PARÁMETROS USADOS EN EL ALGORITMO DE PARTICIONAMIENTO

La selección de los sucesivos puntos p se hizo en forma aleatoria. Se hicieron particiones que dejaran la misma proporción de elementos en ambos núcleos, hasta particiones que dejaran una proporción $10\% - 90\%$, con decrementos de 10. Es decir, se usaron los valores $0.5,\ 0.4,\ 0.3,\ 0.2$ y 0.1 para s. Para cada uno de estos valores de s, se experimentó con valores de 1 hasta 4 para rc.

RESULTADOS

En una primera etapa se trabajó sobre un diccionario español, de 86.061 palabras, orientando los experimentos a obtener los valores óptimos para s y rc. Se eligieron al azar 500 palabras del diccionario, y para cada una de ellas se realizaron búsquedas por rango $(q, r)_d$ usando como radio de búsqueda r los valores $1,\ 2,\ 3$ y 4.

Para cada una de las búsquedas se calculó la proporción $(cmpb + cmpd)/cmpc$, siendo $cmpb$ la cantidad de comparaciones hechas para buscar en el núcleo blando, $cmpd$ la cantidad de comparaciones hechas para buscar en el núcleo duro y $cmpc$ la cantidad de comparaciones hechas para buscar en el diccionario sin particionar. Esta proporción mide el grado de mejora que presenta la técnica sobre el método clásico (sin particionar). Un valor igual a 1 indica que ambos enfoques realizan la misma cantidad de evaluaciones de la función de distancia d . Un valor menor que 1 indica que el enfoque particionado es mejor que el enfoque sin particionar.

La figura 4 muestra los promedios de la proporciones sobre las 500 palabras, usando un GNAT de aridad 110 para el diccionario sin particionar, y un GNAT de aridad 110 o menor en los núcleos. Como puede observarse, independientemente del radio de corte rc, los mejores resultados se obtuvieron para $s = 0.5$, con una reducción en cálculos de distancias realizados de hasta un 35%. Esto significa que, usando la misma cantidad de memoria, podemos ahorrar en tiempo de búsqueda, una cantidad significativa de evaluaciones de distancia. La figura 7 muestra los resultados para $s = 0.5$, con los distintos valores de rc. Notar que la mejor performance se obtiene en $rc = 2$, degradando a medida que rc aumenta.

El próximo paso fué establecer que sucedía al utilizar las aridades óptimas en cada caso. Para esto se indexó el diccionario sin particionar y los núcleos usando aridades $16, 32, 64, 128$ y 256 La figura 5 (izquierda) muestra el comportamiento del GNAT bajo las distintas aridades, para el diccionario sin particionar. Como puede observarse 256 resulta ser la mejor elección para búsquedas de baja selectividad, mientras que 128 es la más adecuada para búsquedas de mayor selectividad.

En cuanto a los núcleos, para cada valor de rc y s analizamos todas las combinaciones de aridades a fin de establecer la combinación óptima. La figura 5 (derecha) muestra un ejemplo para el caso $rc = 2$ y $p = 0.4$. Como conclusión de estos experimentos llegamos a que 128 resulta ser el valor óptimo en ambos núcleos.

Finalmente la figura 6 muestra los resultados usando las aridades más convenientes en cada caso, mateniéndose reducciones de hasta un 35% en cálculos de distancia tal como sucedía en los experimentos anteriores

En cuanto a los tiempos de construcción del índice, el aumentar rc produjo un incremento consi-

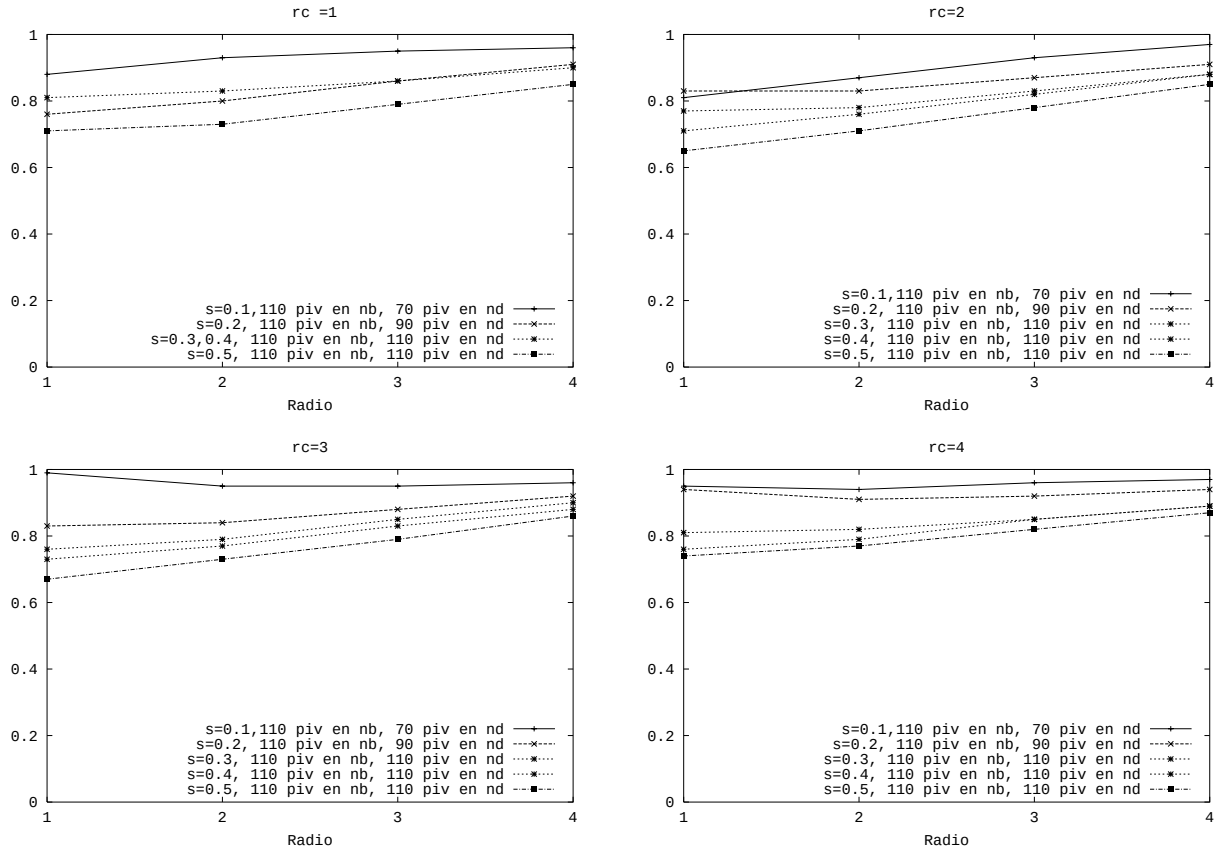

Figura 4: *Resultados para un diccionario Español de 86061 palabras*

derable en la cantidad de puntos p necesitados para encontrar $nd(\mathcal{U}, d)$.

En base a estos resultados se decidió realizar los experimentos posteriores sólo para $rc = 2$.

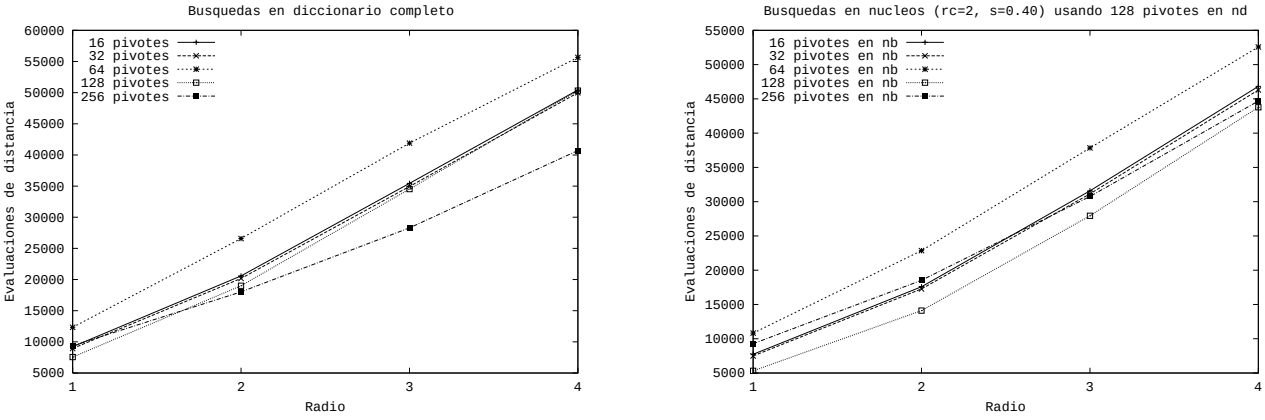

Figura 5: *Comportamiento de GNAT para distintas aridades (izquierda) y determinación de la aridad óptima para cada partición (derecha)*

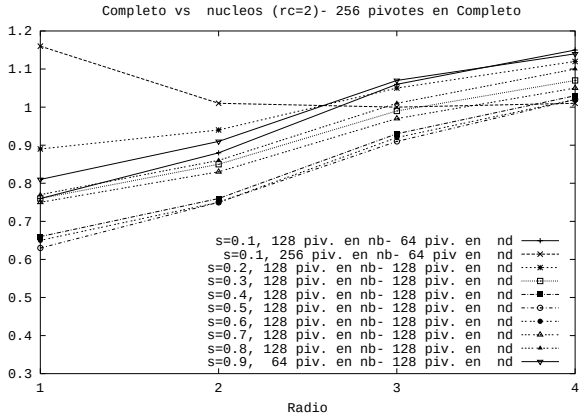

Figura 6: *Comparacion utilizando la mejor aridad en cada uno de los nucleos.*

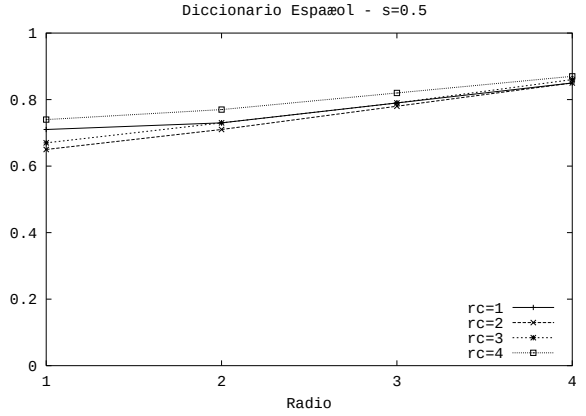

Figura 7: *Resultados para un diccionario Español de 86061 palabras, usando* $s = 0.5$

La figura 8 exhibe los resultados de los experimentos realizados con un diccionario italiano de aproximadamente 116.000 palabras. La figura de la izquierda, muestra los resultados cuando se usa un GNAT de aridad 110 tanto para los núcleos como para el diccionario sin particionar. En este caso, la mejor performance se alcanza para $s = 0.4$, con reducciones en cálculos de distancia de alrededor del 30%.

En este caso, experimentalmente se comprobó que, para indexar el diccionario completo, usar aridad 30 produce mejores resultados que usar aridad 110. La figura de la derecha exhibe los resultados de la comparación contra un GNAT de aridad 30. También aquí logramos reducciones de alrededor del 30% en cálculos de distancia. Notar que en este caso, la cantidad de memoria usada en los núcleos es mayor que la usada para indexar el diccionario completo. Pero aún así, se logra una mejora que no se alcanzaba aumentando la memoria en el diccionario sin particionar.

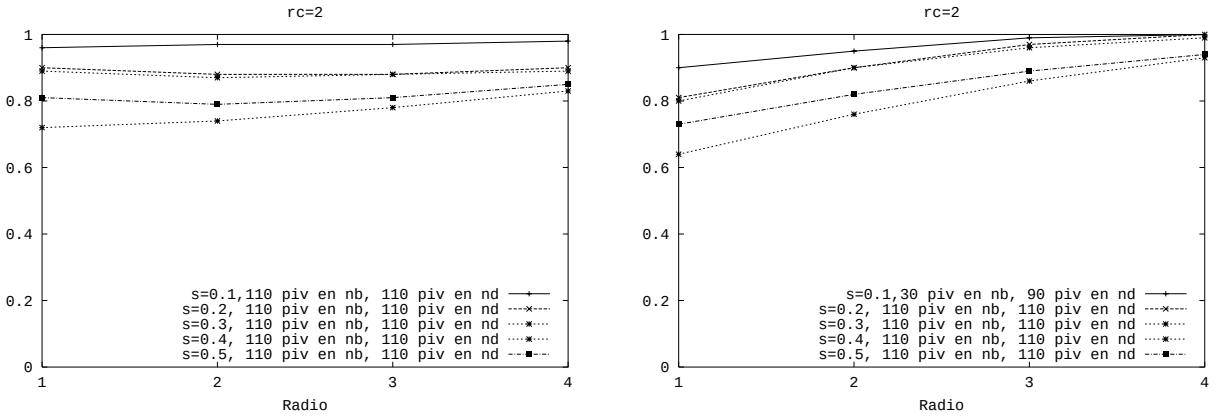

Figura 8: *Resultados para un diccionario Italiano de 116.879 palabras*

Finalmente, la figura 9 muestra los resultados para un diccionario francés de aproximadamente 138.000 palabras. La figura de la izquierda es el resultado de comparar contra un GNAT de aridad 110 en el diccionario sin particionar, y la de la derecha los resultados de comparar contra uno de aridad 50. Esto se debe a que, sobre el diccionario sin particionar un GNAT de aridad 50 tiene mejor desempeño que uno de aridad 110. Se logran reducciones de hasta un 35% para $s = 0.5$.

En este caso, dada la cardinalidad del diccionario, se comprobó que sucedía al aumentar aún mas la aridad del GNAT. En la figura 10 pueden observarse los resultados de la comparación usando un GNAT de aridad 160. Notar que en este caso el grado de mejora no disminuye si aumentamos el radio de búsqueda, algo que no sucedía en los casos anteriores.

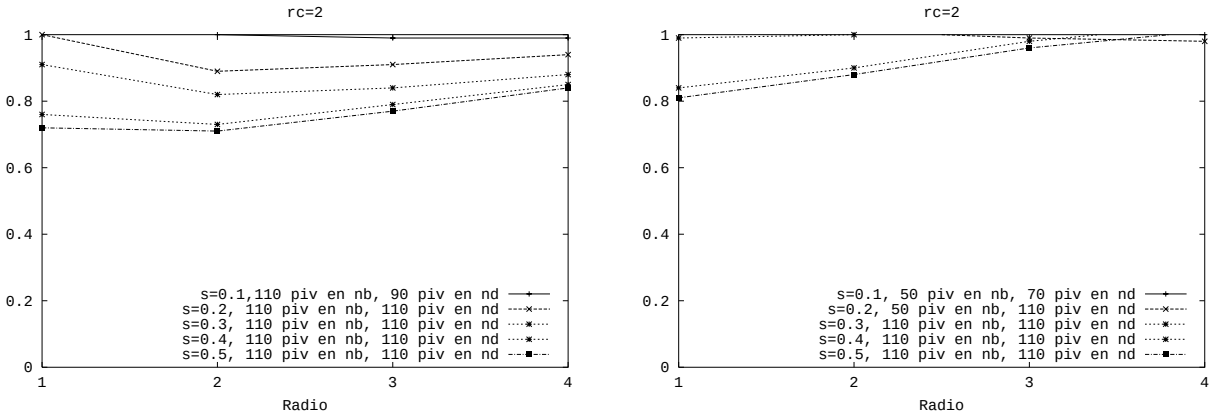

Figura 9: *Resultados para un diccionario Francés de 138.257 palabras*

6 Conclusiones y trabajo futuro

En este trabajo hemos presentado un método de parametrización local que permite particionar el espacio en dos segmentos, de manera tal que a cada segmento se le puedan seleccionar de manera óptima los parámetros correspondientes. La detección de las regiones duras y blandas de un espacio, permite construir índices adecuados a cada región, reduciendo la cantidad de evaluaciones de distancias

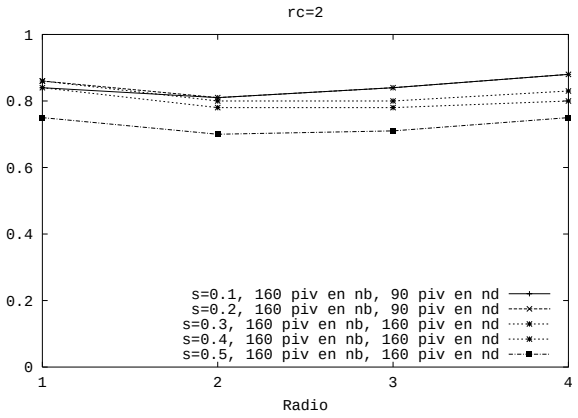

Figura 10: *Resultados para un diccionario Francés de 138.257 palabras*

necesarias para resolver una búsqueda $(q, r)_d$.

Hemos ilustrado el comportamiento de esta estrategia, usando un GNAT para la indexación de los segmentos. En la sección 5 hemos mostrado que se consigue reducir hasta en un 35% las cantidad de evaluaciones de la función d. Estos resultados preliminares son motivantes, permiten anticipar un camino no explorado en la construcción de índices. Los experimentos descritos en el trabajo son limitados (un solo índice, un solo espacio métrico), pero permiten extrapolar el comportamiento en otros espacios métricos. Como se describe en [7], lo relevante para medir la complejidad de la búsqueda en un espacio métrico es el histograma de distancias. El método propuesto utiliza intensivamente este parámetro (con el radio de corte). Este último se puede intercambiar con un percentil del histograma para ser mas aplicable en el caso general.

Observamos que el radio de corte no es independiente del tamaño óptimo de los respectivos segmentos. Para cada radio de búsqueda sería necesario sintonizar estos parámetros. Al parecer las particiones equilibradas (50%) son mas eficientes que las particiones sesgadas. Como trabajo futuro, nos proponemos estudiar posibles optimizaciones a esta estrategia y su comportamiento ante otros algoritmos de indexación.

Con respecto al primer punto, hay varias líneas a seguir:

- Particionar el espacio en una cantidad mayor de segmentos, o construir una jerarquía de particiones.
- Usar técnicas mas costosas en el proceso de detección de los núcleos. El uso de histogramas locales es una técnica sencilla y rápida, técnicas más costosas pueden producir mejores segmentaciones.
- Usar diferentes índices para cada parte. Una possibilidad es usar algoritmos probabilísticos en el núcleo duro y algoritmos exactos en el núcleo blando.

Agradecimientos Queremos agradecer a los árbitros anónimos que con sus comentarios ayudaron a mejorar esta presentación.

Referencias

[1] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), 1991.

[2] J. Bentley. Multidimensional binary search trees used for associative searching. *Comm. of the ACM*, 18(9):509–517, 1975.

[3] J. Bentley. Multidimensional binary search trees in database applications. *IEEE Trans. on Software Engineering*, 5(4):333–340, 1979.

[4] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conference on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

[5] E. Chávez and J. Marroquín. Proximity queries in metric spaces. In R. Baeza-Yates, editor, *Proc. 4th South American Workshop on String Processing (WSP'97)*, pages 21–36. Carleton University Press, 1997.

[6] E. Chávez and G. Navarro. An effective clustering algorithm to index high dimensional metric spaces. In *Proc. 7th International Symposium on String Processing and Information Retrieval (SPIRE'00)*, pages 75–86. IEEE CS Press, 2000.

[7] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[8] Edgar Chávez and Gonzalo Navarro. A probabilistic spell for the curse of dimensionality. In *Proc. 3rd Workshop on Algorithm Engineering and Experiments (ALENEX'01)*, LNCS, 2001.

[9] P. Ciaccia, M. Patella, and P. Zezula. A cost model for similarity queries in metric spaces. In *Proc. 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*, 1998.

[10] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

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