Una Propuesta para la Selección de Pivotes en Indices Métricos *

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

Anabella C. De Battista, Andrés J. Pascal
Departamento de Sistemas de Información
Universidad Tecnológica Nacional
Regional Concepción del Uruguay
{debattistaa, pascalj}@frcu.utn.edu.ar

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 objetos similares. En este trabajo nos hemos centrado en las técnicas basadas en pivotes, las cuales construyen el índice en torno a un grupo de puntos estratégicos de la base de datos denominados pivotes.

El grupo de pivotes utilizado en la construcción del índice no afecta en absoluto la efectividad del mismo, pero es crucial para su eficiencia. Es por esta razón que el tema de selección de un buen grupo de pivotes está siendo ampliamente estudiado. En este artículo presentamos el diseño de dos nuevas técnicas para la selección de un buen grupo de pivotes.

Palabras claves: Bases de Datos, Espacios Métricos, Indices, Selección de Pivotes.

*Este trabajo ha sido parcialmente subvencionado por el proyecto Tecnologías Avanzadas de Bases de Datos (22/F314), Universidad Nacional de San Luis.

1. Introducción

Con la evolución de los sistemas y las tecnologías de información, las bases de datos actuales han incluido la capacidad de almacenar datos no estructurados tales como imágenes, sonido, video, huellas digitales, entre otros. Las búsquedas que son de interés en ese tipo de bases de datos, son aquellas que recuperan objetos similares a un elemento dado. Este tipo de búsqueda se conoce con el nombre de *búsqueda por similitud* y surge en áreas tales como reconocimiento de voz, reconocimiento de imágenes, compresión de texto, recuperación de texto, biología computacional, por nombrar algunas.

El conjunto X de todos los objetos sobre los cuales se puede realizar la búsqueda, junto con una función de distancia d que mide la similitud entre los elementos de X, se denomina *espacio métrico*, y se denota (X, d). La base de datos será un conjunto $U \subseteq X$. La función de distancia $d : X \times X \rightarrow R^+$ debe satisfacer las propiedades que caracterizan a una métrica, a saber:

1. $\forall x, y \in X : d(x, y) \geq 0$ (positividad)
2. $\forall x, y \in X : d(x, y) = d(y, x)$(simetría)
3. $\forall x, y, z \in X : d(x, y) \leq d(x, z) + d(z, y)$ (desigualdad triangular)

Si bien existen distintos tipos de búsquedas por similitud, una de las más comunes es la *búsqueda por rango*. Esta búsqueda, que denotaremos con $(q, r)_d$, consiste en recuperar todos los elementos de U cuya distancia a un elemento q dado no supere un rango de tolerancia r; en símbolos $(q, r)_d = \{u \in \mathcal{U} : d(q, u) \leq r\}$. Nos referiremos a q como elemento de consulta o query.

Hay tres factores que afectan el tiempo necesario para resolver una búsqueda por similitud; por un lado tenemos la cantidad de evaluaciones de la función de distancia d que se realizaron durante el proceso de búsqueda; por otro lado tenemos una cierta cantidad de operaciones adicionales que implican un tiempo extra de CPU; finalmente, tenemos un tiempo de I/O determinado por la cantidad de accesos a memoria secundaria, si es que fuera necesario. En consecuencia, el tiempo total de resolución de una búsqueda se puede calcular de la siguiente manera:

$$T = \#\text{ evaluaciones de } d \times \text{complejidad}(d) + \text{ Tpo de } CPU + \text{ Tpo de } I/O$$

En muchas aplicaciones el cálculo de la función de distancia d es tan costoso que las demás componentes que afectan el tiempo pueden ser despreciadas. En estos casos, la medida de complejidad es la cantidad de evaluaciones de la función de distancia d realizadas al momento de resolver una consulta.

Una búsqueda por similitud puede ser resuelta con $O(n)$ evaluaciones de distancias examinando exhaustivamente la base de datos. Para evitar esta situación, se preprocesa la base de datos por medio de un *algoritmo de indexación* con el objetivo de construir una *estructura de datos o índice*, diseñada para ahorrar cálculos en el momento de resolver una búsqueda. Un algoritmo de indexación se considera eficiente si puede responder una búsqueda por similitud haciendo una cantidad pequeña de cálculos de distancia, sublineal en la cantidad de elementos de la base de datos.

En [7] se presenta un desarrollo unificador de las soluciones existentes en la temática. En dicho trabajo se muestra que todos los enfoques para la construcción de índices en espacios métricos consisten en:

- particionar el espacio en clases de equivalencia.
- indexar las clases de equivalencia.

- durante la búsqueda, usando el índice y la desigualdad triangular, descartar algunas clases y buscar exhaustivamente en las restantes.

La diferencia entre los distintos algoritmos radica en cómo construyen estas clases de equivalencia. Básicamente se pueden distinguir dos enfoques: *algoritmos basados en pivotes* y *algoritmos basados en particiones compactas*. En este trabajo nos hemos centrado sobre algoritmos basados en pivotes [1, 2, 3, 5, 6, 7], dedicándonos al estudio de la problemática de selección de un buen grupo de pivotes.

Este artículo se organiza de la siguiente manera. Comenzamos dando una introducción a algoritmos de indexación basados en pivotes. Luego, en la sección 3, describimos los trabajos existentes sobre la problemática de selección de un buen grupo de pivotes. En la sección 4, presentamos nuestro aporte que consiste en el diseño de dos nuevos algoritmos para la selección de pivotes. Finalizamos el artículo en la sección 5 dando las conclusiones y el trabajo futuro.

2. Algoritmos Basados en Pivotes

Este grupo de algoritmos definen una relación de equivalencia basados en la distancia de los elementos a un conjunto de elementos preseleccionados que llamaremos *pivotes*. Sea $\{p_1, p_2, \ldots, p_k\}$ el conjunto de pivotes, dos elementos son equivalentes si y solo si están a la misma distancia de todos los pivotes:

$$x \sim_{\{p_i\}} y \Leftrightarrow\ d(x, p_i) = d(y, p_i),\ \forall i = 1 \ldots k$$

Gráficamente, cada clase de equivalencia está definida por la intersección de varias capas de esferas centradas en los puntos p_i (ver figura 1, izquierda).

Durante la indexación, se seleccionan k pivotes $\{p_1, p_2, \ldots, p_k\}$, y se le asigna a cada elemento a de la base de datos, el vector o firma:

$$\Phi(a) = (d(a, p_1), d(a, p_2), \ldots, d(a, p_k))$$

Durante la búsqueda, se usa la desigualdad triangular junto con la firma de cada elemento para filtrar objetos de la base de datos sin medir su distancia a la query q. Dada $(q, r)_d$, se computa la firma de la query q, $\Phi(q) = (d(q, p_1), d(q, p_2), \ldots, d(q, p_k))$, y luego se descartan todos aquellos elementos a, tales que para algún pivote p_i se cumple que $\mid d(q, p_i) - d(a, p_i) \mid > r$, es decir:

$$max_{1 \leq i \leq k}\{\mid d(a, p_i) - d(q, p_i) \mid\} = L_\infty(\Phi(a), \Phi(q)) \leq r$$

Los elementos no descartados forman parte de una lista de candidatos, que posteriormente se comparan directamente con la query q. Esto significa que la cantidad total de cálculos de la función de distancia d queda determinada por la cantidad de pivotes k más la cardinalidad de la lista de candidatos.

Notar que la relación de equivalencia definida por un conjunto de k pivotes también puede verse como una proyección al espacio vectorial $\mathbb{R}^k$. La i-ésima coordenada de un elemento es la distancia al i-ésimo pivote. Esto significa que hemos proyectado el espacio métrico original $(\mathcal{X}, d)$ en el espacio vectorial $\mathbb{R}^k$ con la función de distancia L_∞ (ver figura 1).

La mayoría de los índices basados en pivotes seleccionan en forma aleatoria elementos de la base de datos, para formar su grupo de pivotes. Sin embargo, se sabe que la política usada en la selección de pivotes afecta notablemente la performance de la búsqueda [4, 7, 9, 10]. Esto significa que si tenemos dos conjuntos de pivotes del mismo tamaño elegir el mejor de los dos para indexar la base de datos, puede reducir la cardinalidad de la lista de candidatos y, en consecuencia, reducir el tiempo de

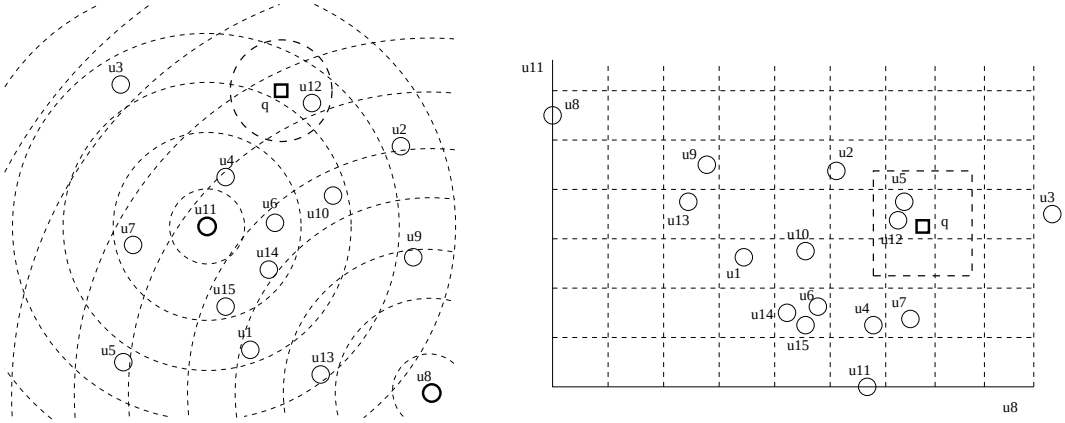

Figura 1: *Un ejemplo de la relación de equivalencia inducida por la intersección de anillos centrados en dos pivotes,* u_8 *y* u_{11} *(izquierda); y su correspondiente transformación en un espacio vectorial de dimensión 2 (derecha). También se ilustra la transformación de una búsqueda* $(q, r)_d$.

búsqueda. Por otro lado, un grupo pequeño de pivotes bien elegidos puede resultar tan eficiente como un grupo de mayor cantidad de pivotes pero elegidos aleatoriamente.

El grupo de pivotes utilizados para la construcción del índice no afecta en absoluto la efectividad del mismo pero es crucial para su eficiencia. Por lo tanto, el tema de selección de un buen grupo de pivotes para indexar un determinado espacio métrico está siendo ampliamente estudiado.

3. Trabajos Relacionados

3.1. Selección Incremental de Pivotes

En [4] se proponen tres técnicas para la selección de un buen grupo de pivotes. Dichas técnicas tratan de maximizar la media μ_D de la distribución de D, donde:

$$D([x],[y])_{\{p_1,\ldots,p_k\}} = max_{1\leq i\leq k}\{\mid d(x,p_i) - d(y,p_i) \mid\}$$

De las tres técnicas allí presentadas, la que muestra un mejor desempeño es *selección incremental*. Este método consiste en tomar una muestra de N elementos de la base de datos y seleccionar como primer pivote p_1 a aquel elemento que tenga el máximo valor para μ_D. El segundo pivote p_2 se elige de otra muestra de N elementos de forma tal que $\{p_1, p_2\}$ tenga el máximo valor para μ_D. Este proceso se repite hasta terminar de elegir los k pivotes necesitados.

En dicho trabajo se muestra que un buen grupo de pivotes tiene dos características básicas:

- los pivotes están alejados unos de otros, es decir, la distancia media entre pivotes es mayor que la distancia media entre elementos tomados al azar del espacio métrico.
- los pivotes están alejados del resto de los elementos del espacio métrico.

Los elementos que tienen estas dos propiedades se denominan *outliers*. También se observa que, si bien buenos pivotes tienen la propiedad de ser outliers, no todos los outliers son eficientes como pivotes. A partir de los resultados experimentales se concluye que los conjuntos de outliers tienen un buen desempeño en espacios vectoriales uniformemente distribuidos, pero tienen una baja performance en espacios métricos generales, aún peor que una selección aleatoria.

3.2. Selección Dinámica del Conjunto de Pivotes

En [8] se presenta un enfoque alternativo al problema de selección de pivotes que consiste en una selección dinámica del conjunto de pivotes y, por consiguiente, del índice sobre el que se resolverá la consulta.

Para lograr esto, en lugar de seleccionar durante la construcción del índice un grupo de pivotes que sea efectivo para todo el espacio métrico, se *selecciona durante la búsqueda un grupo de pivotes que sea efectivo para la query* q. Para ello, se construyen varios índices sobre el espacio con distintos grupos de pivotes (elegidos aleatoriamente); luego, durante una búsqueda $(q, r)_d$ se selecciona aquel índice que sea más adecuado a q de acuerdo al conjunto de pivotes con el que fue construido.

Esta selección dinámica de índices permite además realizar consultas en paralelo, si cada uno de los índices creados se mantiene en distintas máquinas de una red.

Supongamos que se generan M índices de k pivotes cada uno. Si sólo tomamos en cuenta la cantidad de evaluaciones de distancia, esta idea pierde al compararla con la opción de tener un sólo índice de Mk pivotes. Pero si tomamos en cuenta tiempo extra de CPU y tiempo de I/O, la idea de varios índices pequeños aventaja a la opción de un sólo índice con mayor cantidad de pivotes.

En [8] se presentan varias heurísticas para la selección del índice adecuado, ellas son: selección por votos, pivote más cercano, pivote más lejano, menor masa total, pivote de menor masa y votación global. A partir de la evaluación experimental de las mismas se muestra que las de mejor desempeño son las heurísticas *pivote de menor masa y votación global*:

PIVOTE DE MENOR MASA:

Dado un pivote p y una búsqueda $(q, r)_d$ sabemos que los elementos que no pueden ser eliminados por p son aquellos x tales que $d(x, p) \in [d(p, q) - r, d(p, q) + r]$.

Consideremos el histograma local del pivote p (ver figura 2). Este histograma permite ver la distribución de los elementos u del espacio respecto de p. El eje x representa los distintos valores para $d(u, p)$ y el eje y muestra la cantidad de elementos del espacio que están a una determinada distancia de p. En esta figura la cantidad de elementos que no podrán eliminarse ante una búsqueda $(q, r)_d$, corresponde al área sombreada. Llamaremos a esta zona la *masa* de p para la query q, y la denotaremos con $m(p, q)$.

Suponiendo que se han generado M índices de k pivotes cada uno, la heurística de selección de pivote de menor masa selecciona aquel índice que contenga el pivote de menor masa para una query q dada. Si denotamos con p_{ij} al $j-$ésimo pivote del índice i, podemos expresar lo anterior de la siguiente manera

- $(\forall i)_{1 \leq i \leq M}\ (\forall j)_{1 \leq j \leq k}$: calcular $m(p_{ij}, q)$
- Seleccionar aquel índice i tal que $m(p_{ij}, q)$ sea mínima para algún j, con $1 \leq j \leq k$.

Dado que durante la construcción de un índice se calculan las distancias de todos los elementos de la base de datos a todos los pivotes, en ese momento es posible obtener y almacenar el histograma local de todos los pivotes. De esta manera, durante la búsqueda, no hay costo adicional de tiempo por cálculo de histogramas.

VOTACIÓN GLOBAL :

Todas las heurísticas planteadas (selección por votos, pivote más cercano, pivote más lejano, menor masa total y pivote de menor masa) permiten no sólo seleccionar un índice sino ordenar todos los índices según la conveniencia para una búsqueda $(q, r)_d$.

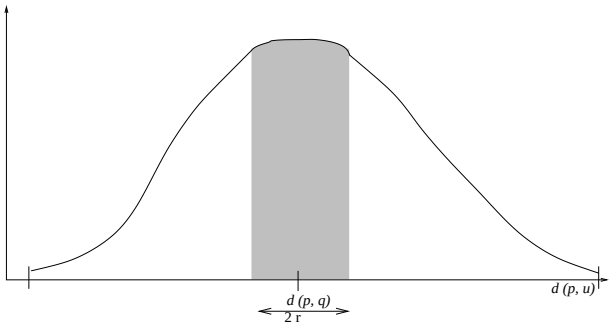

Figura 2: *Histogramas local de un pivote p. El área sombreada representa los elementos que no pueden ser eliminados por p ante una búsqueda $(q, r)_d$*

Dada una query q y una heurística h, puede suceder que h cometa un error y, en consecuencia, el índice que resulta seleccionado no es el óptimo para q. Esto significa que si miramos los índices ordenados según la visión de h para q , el índice óptimo queda desplazado del primer lugar.

La técnica de votación global, mediante la asignación de votos a los índices, trata de captar la información dada por cada una de las heurísticas con el objetivo de bajar la probabilidad de error en la selección del índice.

Sea $\langle I_{j_1}, I_{j_2}, \ldots, I_{j_M} \rangle$ la secuencia de índices ordenados por una heurística h de acuerdo a su conveniencia para la query q. Luego, I_{j_s} recibe una cantidad de votos que depende de la probabilidad de que la técnica considerada desplace a la posición s el índice óptimo para q.

Dada una query q, se realiza el proceso de votación con las heurísticas selección por votos, pivote más cercano, pivote más lejano, menor masa total y pivote de menor masa. El índice que finalmente resulta electo es aquel que recibe la mayor cantidad de votos totales.

En [8] también se analizan las características que presentan los pivotes de los índices más competitivos, siendo la observación más importante que *los mejores índices son aquellos cuyo grupo de pivotes tienen una mayor varianza*.

4. Nuevos Algoritmos para la Selección de Pivotes

Basándonos en los resultados que se muestran en [4] y [8] diseñamos dos nuevas técnicas para la selección de un buen grupo de pivotes. Estas técnicas permiten elegir elementos de la base de datos que tienen característica apropiadas para ser considerados buenos pivotes. Explicamos a continuación cada una de ellas y realizamos el análisis de complejidad de las mismas, midiendo la cantidad de evaluaciones de distancia.

4.1. Selección incremental maximizando la varianza

Tal como se muestra en [4], la selección incremental maximizando μ_D resulta efectiva. Por otro lado en [8] se señala que los índices más competitivos tienen pivotes que maximizan la varianza. La política de selección que proponemos combina estos resultados permitiendo realizar una selección incremental maximizando la varianza.

Básicamente el proceso es el siguiente. El primer pivote p_1 se elije de una muestra de N elementos de manera tal que ese único pivote maximice la varianza σ_D. Habiendo fijado p_1, el segundo pivote p_2 se elije de otra muestra de N elementos de manera tal que el conjunto $\{p_1, p_2\}$ maximice la varianza σ_D. Este proceso se repite hasta completar los k pivotes requeridos.

```
Seleccionar_ Máxima_Varianza( in k);
1. ℙ = ∅
2. Para i = 1 hasta k hacer
3.     p_i = Seleccionar_ p_i(ℙ)
4.     ℙ = ℙ ∪ p_i
5. Fin Para i
9. Retornar ℙ


Seleccionar_ p_i( in ℙ);
1. Elegir en forma aleatoria el conjunto {u_1, u_2, ..., u_N}
2. máx = −∞
3. Para j = 1 hasta N hacer
4.     Si σ_{D_{ℙ ∪ u_j}} > máx entonces
5.         máx = σ_{D_{ℙ ∪ u_j}}
6.         p_i = u_j
7.     Fin Si
8. Fin Para j
9. Retornar p_i
```

Figura 3: *Algoritmo de selección incremental maximizando la varianza*

La figura 3 muestra este proceso de selección. El algoritmo sólo necesita como parámetro de entrada la cantidad k de pivotes a seleccionar y retorna como salida el conjunto $\mathbb{P}$ que contendrá los k pivotes seleccionados.

En el paso i, el conjunto $\mathbb{P}$ contendrá los $i-1$ pivotes ya seleccionados y se procederá a seleccionar el $i-$ésimo pivote. Para la selección del pivote p_i se requiere calcular $\sigma_{D_{\{\mathbb{P} \cup u_j\}}}$ para los distintos u_j. Si el espacio tiene n elementos, el cálculo exacto de esta varianza implicaría $O(n^2)$ evaluaciones de distancias. Siguiendo las ideas dadas en [4], podemos evitar este costo estimando $\sigma_{D_{\{p_1,\ldots,p_i\}}}$ de la siguiente manera:

- Seleccionar en forma aleatoria A pares de elementos $(a_1, a'_1), (a_2, a'_2), \ldots, (a_A, a'_A)$.
- Para cada par (a_t, a'_t) calcular

$$D_t = D([a_t], [a'_t])_{\{p_1,\ldots,p_i\}} = max_{1 \leq j \leq i}\{ \mid d(a_t, p_j) - d(a'_t, p_j) \mid \}$$

  Esto produce como resultado el conjunto de valores $\{D_1, D_2, \ldots, D_A\}$.

- Estimar la media $\mu_{D_{\{p_1,\ldots,p_i\}}}$ de la siguiente forma:

$$\mu_{D_{\{p_1,\ldots,p_i\}}} = \frac{1}{A} \sum_{1 \leq j \leq A} D_j$$

- Estimar la varianza $\sigma_{D_{\{p_1,\ldots,p_i\}}}$ de la siguiente forma:

$$\sigma_{D_{\{p_1,\ldots,p_i\}}} = \frac{1}{A} \sum_{1 \leq j \leq A} (D_j - \mu_{D_{\{p_1,\ldots,p_i\}}})^2$$

Una observación importante es que el mismo conjunto de A elementos debe usarse en todo el proceso de selección de los k pivotes, caso contrario los resultados obtenidos en cada estimación no serían comparables porque corresponderían a distintas muestras de elementos.

Dado que hay A pares de elementos que se comparan con los i pivotes, esta estimación de la varianza requiere $2Ai$ evaluaciones de distancia. Como A es una constante del algoritmo y el mayor valor posible para i es k, tenemos que lo anterior es $O(k)$.

Para la selección de un pivote, se realiza esta estimación N veces, lo que implica $NO(k)$ evaluaciones de distancia, que es $O(k)$ (N es una constante del algoritmo). Como en total se seleccionan k pivotes, podemos concluir que la complejidad del proceso de selección maximizando la varianza tiene una complejidad $O(k^2)$.

4.2. Selección incremental por votos

Esta técnica realiza una adaptación de las heurísticas más prometedoras presentadas en [8] a fin de poder realizar una selección estática de pivotes.

Para seleccionar un grupo de k pivotes, se realizan los siguientes pasos:

- Se seleccionan M grupos de h pivotes cada uno (con $h < k$).
- Se selecciona un grupo de A queries.
- Para cada query, se calcula cuál de los M grupos contiene el pivote de menor masa para la query. A ese grupo se le asigna un voto.
- El grupo que resulta con mayor cantidad de votos pasa a formar parte del conjunto de pivotes.
- Se repiten los pasos anteriores hasta completar los k pivotes requeridos.

En la figura 4 se muestra el pseudocódigo de este proceso. El proceso *votar (in q)* , toma como entrada un query y selecciona aquel grupo g que contiene el pivote de menor masa para q, asignándole un voto a dicho grupo. Cuando cada una de las A queries ha realizado el proceso de votación, se agregan al conjunto $\mathbb{P}$ los pivotes del grupo más votado. Cabe señalar que M, h y A son constantes para este algoritmo.

Veamos ahora cuál es el costo de este proceso de selección, medido en cantidad de evaluaciones de distancias. El primer punto donde se realizan evaluaciones de distancia es en el momento de calcular los histogramas locales de los p_{ij} (paso 6). Si en el espacio métrico hay n elementos, el cálculo de los histogramas implicarían $Mhn = O(n)$ evaluaciones de distancia. Para bajar este costo, los histogramas de los pivotes se pueden aproximar tomando sólo una muestra de elementos del espacio [7]. De esta manera, el cálculo de los histogramas requiere sólo Mh evaluaciones de distancia.

El proceso de votación requiere calcular la masa de q a cada uno de los Mh pivotes, lo que implica calcular la distancia de q a cada uno de esos pivotes. Por lo tanto el proceso de votación requiere Mh evaluaciones de distancia. Como la votación se invoca con cada una de las A queries, en total tenemos MhA evaluaciones de distancia.

Como los pasos descriptos se realizan en total k/h veces, tenemos que la cantidad total de evaluaciones de distancias hechas en este proceso de selección es $k/h(Mh + AMh)$ que es $O(k)$.

```
Seleccionar_por_Votos (in k);
1. Elegir en forma aleatoria un conjunto de A queries {q_1, q_2, ..., q_A}
2. P = Ø
3. Repetir
4.    Para i = 1 hasta M hacer
5.       Para j = 1 hasta h hacer
5.          Elegir en forma aleatoria p_ij
6.          Calcular el histograma local de p_ij
5.       Fin Para j
6.    Fin Para i
7.    Para t = 1 hasta A hacer
8.       votar (q_t)
9.    Fin Para t
10.   Sea i el grupo con mayor cantidad de votos:
          P = P ∪ {p_i1, ..., p_ih}
11. Hasta que |P| = k
12. Retornar P


Votar (in q);
1.    mín = +∞ ; g = 0
2.    Para i = 1 hasta M hacer
3.       Para j = 1 hasta h hacer
4.           Si m(p_ij, q_t) < mín entonces
5.                mín = m(p_ij, q_t)
6.                g = i
7.             Fin Si
8.        Fin Para j
9.     Fin Para i
10.    votos(g)++
```

Figura 4: *Algoritmo de selección por votos, minimizando la masa*

5. Conclusiones y Trabajo Futuro

Basándonos en los resultados que se muestran en [4] y [8] hemos diseñamos dos nuevas técnicas para la selección de un buen grupo de pivotes. Estas técnicas permiten, con un bajo costo, elegir elementos de la base de datos que tienen característica apropiadas para ser considerados buenos pivotes.

Los dos algoritmos presentados tienen constantes cuyos valores no hemos especificado. En el caso de selección por máxima varianza existen dos constantes N y A cuyos valores pueden ser heredados de los valores que se recomiendan en [4] para la selección incremental maximizando la media. Algo similar se puede realizar con las constantes M, h y A de la selección por votos, tomado los valores recomendados en [8].

Como trabajo futuro nos proponemos estudiar experimentalmente el comportamiento de estas técnicas sobre espacios métricos reales. Nos proponemos estudiar dos tópicos. Por una lado analizaremos experimentalmente si los valores para N y A recomendado en [4], y los valores para M, h y A recomendados en [8] son adecuados para estas nuevas políticas de selección. En caso de que no lo sean, estableceremos experimentalmente los valores más adecuados. Por otra parte analizaremos el efecto de estas políticas de selección sobre la eficiencia de los índices basados en pivotes.

Referencias

[1] R. Baeza-Yates. Searching: an algorithmic tour. In A. Kent and J. Williams, editors, *Encyclopedia of Computer Science and Technology*, volume 37, pages 331–359. Marcel Dekker Inc., 1997.

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

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

[4] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. In *Proc. of the XXI Conference of the Chilean Computer Science Society (SCCC'01)*, pages 33–40. IEEE CS Press, 2001.

[5] E. Chávez and K. Figueroa. Faster proximity searching in metric data. In *Proceedings of MICAI 2004. LNCS 2972, Springer*, Cd. de México, México, 2004.

[6] E. Chávez, J. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2):113–135, 2001.

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

[8] E. Chávez and N. Herrera. Selección dinámica de índices métricos para consultas de proximidad. In *Actas del X Congreso Argentino de Ciencias de la Computación (CACIC'04)*, pages 389–400, Buenos Aires, Argentina, 2004.

[9] A. Faragó, T. Linder, and G. Lugosi. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):957–962, 1993.

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