UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

INVESTIGACIÓN DE TÉCNICAS DE SELECCIÓN DE PIVOTES PARA ALGORITMOS DE BÚSQUEDA EN ESPACIOS MÉTRICOS

BENJAMIN EUGENIO BUSTOS CÁRDENAS

2001

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

INVESTIGACIÓN DE TÉCNICAS DE SELECCIÓN DE PIVOTES PARA ALGORITMOS DE BÚSQUEDA EN ESPACIOS MÉTRICOS

BENJAMIN EUGENIO BUSTOS CÁRDENAS

COMISIÓN EXAMINADORA		NOTA (nº)	CALIFICACIONES (Letras)	FIRMA
PROFESOR GUÍA SR. GONZALO NAVARRO	:	…….	…….	……...
PROFESOR CO-GUÍA SR. RICARDO BAEZA-YATES	:	…….	…….	…….
PROFESOR INTEGRANTE SR. PATRICIO POBLETE	:	…….	…….	…….
NOTA FINAL EXAMEN DE TÍTULO	:	…….	…….	…….

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

SANTIAGO DE CHILE
ENERO 2001

INVESTIGACIÓN DE TÉCNICAS DE SELECCIÓN DE PIVOTES PARA ALGORITMOS DE BÚSQUEDA EN ESPACIOS MÉTRICOS

En la actualidad existen muchas aplicaciones prácticas en las cuales se tiene una base de datos, no necesariamente estructurada, de donde se requiere extraer elementos "parecidos" a una consulta. Para realizar dicha búsqueda en proximidad es necesario comparar los elementos de la base de datos con la consulta realizada. En general dicha comparación es computacionalmente costosa, por lo que revisar todos los elementos de la base de datos, es decir una búsqueda exhaustiva, resulta ser demasiado caro.

Si es posible definir entre los elementos de la base de datos una función de distancia que respete la desigualdad triangular, entonces se dice que los elementos conforman un *espacio métrico*. En este tipo de espacios es posible responder consultas sin recurrir a la búsqueda exhaustiva. Existen muchos algoritmos de búsqueda en espacios métricos, de los cuales algunos se basan en el uso de *pivotes*, que son elementos distinguidos dentro de la base de datos, y que permiten reducir el número de comparaciones entre elementos necesarias para responder las consultas.

Por lo general dichos pivotes son elegidos de manera aleatoria dentro del conjunto de elementos, pero se sabe que se pueden lograr mejores resultados eligiéndolos adecuadamente. Se han propuesto algunas heurísticas de selección, pero por lo general no se fundamentan formalmente, sólo funcionan en los espacios métricos particulares en donde se han probado y no son extensibles a otros casos prácticos. El objetivo de esta memoria es presentar un criterio de eficiencia para elegir los pivotes basado en propiedades intrínsecas del conjunto de elementos, como es el caso de su histograma de distancias, y a partir de dicho criterio definir técnicas de selección de pivotes.

En la presente memoria se proponen distintas técnicas de selección, basadas en el criterio de eficiencia definido, y se compara su rendimiento en espacios vectoriales aleatorios, que son un caso particular de espacio métrico. De los resultados obtenidos se concluye que el *método incremental de selección de pivotes* es la técnica con la cual se obtienen mejores resultados en la práctica.

A continuación se muestra cómo dicha técnica de selección es directamente aplicable a varios algoritmos de búsqueda en proximidad y se analizan las características de un buen conjunto de pivotes, de lo cual se concluye que los buenos pivotes son elementos aislados entre sí y del resto de los elementos del conjunto. A dichos elementos se les denomina *outliers*. Sin embargo, se muestra que si bien los buenos pivotes son *outliers* no siempre los *outliers* son buenos pivotes, aunque en los experimentos que se realizan en espacios vectoriales aleatorios los *outliers* presentan un mejor rendimiento que al elegir pivotes con el método incremental.

Finalmente, se prueba la técnica de selección de pivotes en distintos casos de la vida real, donde se comprueba que efectivamente el rendimiento del algoritmo de búsqueda mejora consistentemente al elegir los pivotes utilizando el método incremental. Esto muestra que para garantizar resultados buenos en distintos tipos de espacios métricos no es suficiente obtener resultados positivos en casos particulares, sino que es necesaria una fundamentación teórica a la hora de proponer heurísticas. Esto es especialmente importante cuando se trabaja en un contexto de gran diversidad como los espacios métricos.

Agradecimientos

Primero quiero agradecer a Cecilia Cárdenas y a su familia por haberme acogido en su hogar durante mis años de estudio en la Universidad. Sin su ayuda y su apoyo me hubiera resultado mucho más difícil realizar mis estudios, y sé que es en gran parte debido a ellos que he podido lograr y superar las metas que me propuse durante este largo camino.

También quiero agradecer al proyecto FONDECYT 1-000929 por el apoyo económico entregado para la realización de esta investigación; a mi profesor guía, Gonzalo Navarro, por toda su orientación durante el desarrollo de esta memoria; a Edgar Chávez, por sus valiosos comentarios, a mis padres y a todos mis amigos y amigas que me dieron su apoyo en los momentos difíciles.

Por último, un agradecimiento especial a *katherine*, quien con sus 128 Mb de RAM y sus 450 Mhz de velocidad de reloj de CPU batalló incansablemente en contra de la maldición de la dimensionalidad.

Benjamin Bustos

Santiago, Enero de 2001

Indice de contenidos

Indice de figuras

Indice de ecuaciones

Indice de gráficos

Indice de tablas

1. Introducción

Muchas aplicaciones computacionales necesitan buscar información dentro de una base de datos. Dicha búsqueda se puede realizar a través de una **búsqueda exacta**, esto es, se busca un elemento cuyo identificador corresponda exactamente a una llave de búsqueda definida, o a través de una **búsqueda en proximidad**, en donde lo que interesa es encontrar un elemento de la base de datos **suficientemente cercano** a dicha llave de búsqueda.

En la mayoría de los casos, las bases de datos tradicionales trabajan en el enfoque de búsqueda exacta. Los elementos de la base de datos se definen como registros o datos estructurados, los cuales poseen un **identificador único**. Al realizar una búsqueda, se compara la llave de búsqueda con el identificador de cada registro, terminando la búsqueda cuando la llave y el identificador coinciden exactamente. El resultado de la operación es aquel elemento de la base de datos que posee dicho identificador, si es que éste existe.

Ultimamente han aparecido muchas bases de datos que contienen información no estructurada, como por ejemplo imágenes, sonidos, texto libre, etc., en donde **no se puede definir claramente un identificador para cada registro de la base de datos**. Aunque sea posible estructurar dicha información de la manera clásica, no necesariamente la búsqueda de elementos en dicha base de datos querrá ser realizada solamente comparando la llave con el identificador del registro, sino que podría ser necesario realizarla comparando la llave con cualquiera de los campos del registro. Además, no necesariamente se desearía consultar por algún elemento específico perteneciente a la base de datos, como en el caso de la búsqueda exacta, sino por algún elemento que fuera **parecido** a la llave de búsqueda, lo que corresponde al concepto de la búsqueda en proximidad.

Las consultas correspondientes a una búsqueda en proximidad tienen muchas aplicaciones prácticas tales como:

- Consulta por contenido en objetos multimedia: Reconocimiento de rostros, reconocimiento de huellas dactilares, reconocimiento de voz, etc., por ejemplo en el uso de dispositivos biométricos que se utilizan para identificar a una persona en particular. A no ser que sean copias digitales, es muy improbable que dos imágenes distintas del mismo objeto multimedia sean exactamente iguales, por lo que realizar una búsqueda exacta no tiene mucho sentido [1].

- Corrección ortográfica de textos: Dada una palabra de un texto con error ortográfico encontrar la(s) palabra(s) candidata(s) que corrija(n) el error [2].

- Similitud de documentos: Dado un documento, encontrar en una base de datos los documentos más "similares" a éste, donde la similitud de documentos puede definirse, por ejemplo, como la distancia coseno entre dichos documentos.

- Biología computacional: estudio de secuencias de ADN. Estas pueden ser modeladas como textos, y el problema es encontrar secuencias de caracteres dentro de una secuencia mayor, tomando en consideración que una ocurrencia exacta es muy improbable [2,3].

En la actualidad existen muchos algoritmos que resuelven el problema de la búsqueda en proximidad basándose en un enfoque de caja negra, donde la única característica relevante del conjunto de elementos es que definen un **espacio métrico**, es decir, existe una función d sobre el conjunto de los elementos que cumple con las características de una **distancia**: positividad estricta, simetría, reflexividad y desigualdad triangular. Intuitivamente, la función d permite decir qué tan cercano está un elemento del conjunto a otro elemento válido cualquiera.

Los algoritmos de búsqueda en proximidad utilizan esta función d para responder consultas del tipo: ¿Qué elementos de la base de datos se encuentran "cerca" de un cierto elemento x?, en donde el elemento x no necesariamente pertenece a la base de datos, y el concepto de cercanía se puede pensar intuitivamente como un radio de tolerancia. A este tipo consulta se le denomina **consulta por rango**. Para responderla, los algoritmos de búsqueda en proximidad utilizan distintos métodos de indexación del conjunto, con lo que intentan minimizar el número de evaluaciones de la función de distancia, ya que en general **la evaluación de esta función d es computacionalmente costosa**. Por ejemplo, existe toda una rama de algoritmos de búsqueda aproximada que indexan el conjunto utilizando **pivotes** [4,5,6,7,8,9,10], donde un pivote se define como un elemento distinguido del conjunto a indexar, y se elige de entre los elementos de la base de datos utilizando alguna técnica de selección.

Uno de los grandes obstáculos en el diseño de algoritmos de búsqueda en proximidad es la llamada **maldición de la dimensionalidad** [6,11,12,13]. Si se considera el conjunto de elementos como puntos en un espacio vectorial, que es un caso particular de un espacio métrico, su dimensión corresponde al número de coordenadas que tienen los puntos que lo componen. Todos los algoritmos de búsqueda en proximidad empeoran su rendimiento cuando aumenta la dimensión del conjunto, hecho conocido como maldición de la dimensionalidad. En [11] se muestra que el concepto de dimensión de un espacio vectorial se puede

extender a espacios métricos arbitrarios, utilizando el concepto de **dimensión intrínseca** (ver sección 4), y que la maldición de la dimensionalidad se mantiene en dichos espacios. Esto implica que a medida que aumenta la dimensión intrínseca del espacio métrico los algoritmos basados en pivotes necesitan una mayor cantidad de éstos para mantener, en cierta medida, su rendimiento, y que de todas maneras el rendimiento empeora cuando crece la dimensión del espacio [11].

Debido a esto, los algoritmos basados en pivotes necesitan cantidades enormes de memoria si el espacio a indexar es de alta dimensionalidad, lo que hace que muchas veces la memoria sea insuficiente para manejar el problema. En la práctica, para espacios con dimensión intrínseca mayor o igual a 20 se considera el problema como intratable.

En [11] se muestra que el número óptimo de pivotes es $\Theta(\log(n))$ si se eligen al azar, donde n es el número de elementos de la base de datos. Por otro lado, en [12] se demuestra formalmente que si la dimensión es constante y a través de una selección adecuada, con una cantidad fija de pivotes el costo de la búsqueda en proximidad es $O(1)$, lo que muestra que la forma en que los pivotes son escogidos afecta el rendimiento del algoritmo. Esto implica que una mejora en la forma de elegir los pivotes permitiría usar menos memoria, y por lo tanto se podrían abordar nuevas aplicaciones en donde los espacios métricos posean una alta dimensionalidad. Lamentablemente, la demostración del teorema no es constructiva y no indica cuál es la mejor manera de seleccionar los pivotes.

Las técnicas de selección de pivotes es tema de estudio en la actualidad, y se sabe muy poco al respecto. En [11] se describen algunos métodos de selección de pivotes, pero la mayoría de estos métodos son heurísticas con argumentos vagos. En general, casi todos los algoritmos basados en pivotes los eligen de manera aleatoria dentro del conjunto de elementos.

El problema que aborda esta memoria es encontrar una manera formal de selección de pivotes que mejore el rendimiento de los **algoritmos de búsqueda en proximidad basados en pivotes**, es decir, que mejore el rendimiento del algoritmo con respecto a elegir los pivotes al azar, además de encontrar argumentos formales que expliquen por qué un conjunto de pivotes es mejor que otro.

En el capítulo 2 de esta memoria se introducen algunos conceptos básicos necesarios para el entendimiento del resto del texto. En el capítulo 3 se indican algunos trabajos relacionados con algoritmos de búsqueda en proximidad basados en pivotes, para mostrar cuál es el estado del arte en el tema. En el capítulo 4 se introduce el concepto de dimensión intrínseca y se explica el por qué a medida que ésta crece se produce el fenómeno de la maldición de la dimensionalidad; también en este capítulo se discuten

ciertos argumentos a tener en cuenta en una adecuada selección de pivotes para indexar un conjunto de elementos. En el capítulo 5 se formaliza el criterio de selección de pivotes, y se discuten distintas técnicas de selección, su costo asociado y ventajas comparativas entre ellas. En el capítulo 6 se muestran los resultados de los experimentos realizados utilizando las distintas técnicas de selección de pivotes, mostrando cual de ellas es la que funciona mejor. En el capítulo 7 se muestra una aplicación práctica de la técnica de selección elegida, ocupando algunos casos de la vida real. Finalmente, en el capítulo 8 se analizan algunas conclusiones que se obtienen del trabajo realizado y se dan recomendaciones generales sobre la elección de pivotes en algoritmos de búsqueda en espacios métricos.

2. Conceptos básicos

En esta sección se introducen los conceptos de espacio métrico y funciones de distancia para el caso particular de los espacios vectoriales, se define lo que es una consulta por rango en un espacio métrico y se explica el algoritmo básico de búsqueda en proximidad usando pivotes.

2.1 Espacios métricos

Un espacio métrico se define como un par (E, d) con $E \subset U$, donde U es el universo de los elementos válidos del espacio, E es un subconjunto finito de U que representa la base de datos de la cual se quiere extraer información, y d es una función $d : UxU \rightarrow \Re$ que cumple las siguientes propiedades:

- $\forall x, y \in U, d(x, y) \geq 0$ (positividad).
- $\forall x, y \in U, d(x, y) = d(y, x)$ (simetría).
- $\forall x \in U, d(x, x) = 0$ (reflexividad).
- $\forall x, y \in U, x \neq y \Rightarrow d(x, y) > 0$ (positividad estricta).
- $\forall x, y, z \in U, d(x, z) \leq d(x, y) + d(y, z)$ (desigualdad triangular).

La función d se denomina **distancia** o **métrica** del espacio E.

2.2 Funciones de distancia en espacios vectoriales

Un ejemplo de métricas, en el caso de los espacios vectoriales, es la familia de distancias L_s o distancias de Minkowski, definida en la ecuación 2-1.

$$L_s((x_1,..,x_n),(y_1,..,y_n)) = (\sum_{i=1}^{n} |x_i - y_i|^s)^{\frac{1}{s}}, s \geq 1$$

Ecuación 2-1

Algunos casos particulares de esta familia de distancias son:

- L_1, conocida como "distancia Manhattan" (ecuación 2-2).

$$L_1((x_1,..,x_n),(y_1,..,y_n)) = \sum_{i=1}^{n} |x_i - y_i|$$

Ecuación 2-2

- L_2, conocida como "distancia Euclidiana" (ecuación 2-3).

$$L_2((x_1,..,x_n),(y_1,..,y_n)) = \sqrt{(\sum_{i=1}^{n} |x_i - y_i|^2)}$$

Ecuación 2-3

- L_∞, conocida como "distancia del máximo", que corresponde al límite cuando *s* tiende a infinito (ecuación 2-4).

$$L_\infty((x_1,..,x_n),(y_1,..,y_n)) = \max_{i=1}^{n}(|x_i - y_i|)$$

Ecuación 2-4

2.3 Consulta por rango en un espacio métrico

Dado un espacio métrico (E,d), un elemento $q \in U$ y un radio de tolerancia $r > 0, r \in \Re$, se define una **consulta por rango** $(q,r)_d$ sobre el espacio métrico (E,d) como:

$$(q,r)_d = \{x \in E, d(q,x) \leq r\}$$

Ecuación 2-5

La figura 2-1 muestra un ejemplo de una consulta por rango en un espacio vectorial de dimensión 2:

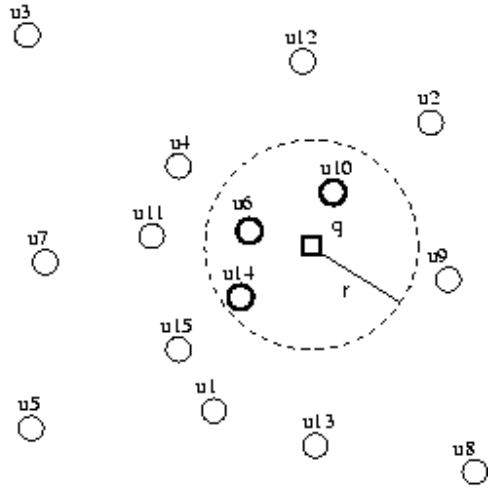

Figura 2-1: Ejemplo de consulta por rango

Se observa en la figura que para dicha consulta $(q,r)_d$ el resultado es el conjunto de elementos $\{u6, u10, u14\}$.

En adelante, se referirá a una consulta por rango (ecuación 2-5) como la consulta (q,r), $q \in U$, $r \in \Re$.

2.4 Algoritmo básico de búsqueda en proximidad usando pivotes

Dada una consulta (q,r) y un conjunto de k pivotes $p_i \in E$, por la desigualdad triangular se tiene que $d(p_i,x) \le d(p_i,q) + d(q,x)$, con $x \in E$, y de la misma forma se tiene que $d(p_i,q) \le d(p_i,x) + d(q,x)$. De las inecuaciones anteriores se obtiene que una cota inferior para la distancia entre q y x es $d(q,x) \ge |d(p_i,x) - d(p_i,q)|$. Como los elementos x que interesan son aquellos en donde $d(q,x) \le r$, entonces se pueden excluir todos los elementos que no cumplan la condición de la ecuación 2-6:

$$|d(q,p_i) - d(x,p_i)| \le r, \forall i = 1..k$$

Ecuación 2-6

Si el espacio E posee n elementos, se construye un índice con las nk distancias $d(x, p_i)$, y por lo tanto al momento de realizar la consulta (q, r) solo es necesario calcular las k distancias $d(q, p_i)$. A esto se le denomina **complejidad interna** de la consulta (q, r). Es fácil ver que mientras mayor sea el conjunto de pivotes, mayor será la complejidad interna de la consulta.

Los elementos x no descartados por la condición de la ecuación 2-6 deben verificarse directamente con q y comprobar si verdaderamente se cumple la condición descrita en la ecuación 2-5. A este cálculo de distancias adicionales se le denomina **complejidad externa** de la consulta (q, r).

Por lo tanto, la complejidad total de la consulta (q, r) es la suma de sus complejidades interna y externa. En [11] se muestra a través de resultados empíricos que existe un número óptimo k^* de pivotes que minimiza la complejidad total de la consulta (q, r), y lo que se busca con las técnicas de selección de pivotes es disminuir dicho k^* con respecto a pivotes elegidos al azar, y que la complejidad total de la consulta también sea menor en el nuevo óptimo.

2.5 Teorema de Chebyschev

Sea una variable aleatoria X cualquiera. La probabilidad que X tome un valor dentro de k desviaciones estándar de la media es al menos $1-\frac{1}{k^2}$ [14]. Es decir:

$$\Pr(\mu - k\sigma < X < \mu + k\sigma) \geq 1 - \frac{1}{k^2}$$

Ecuación 2-7

3. Algoritmos de búsqueda en espacios métricos

En la actualidad existen muchos algoritmos de búsqueda que indexan, de alguna manera, un espacio métrico. Dentro de estos algoritmos, hay toda una rama de ellos que utilizan pivotes para realizar la indexación del espacio. Ejemplos de este tipo de algoritmos son:

- Burkhard-Keller Tree: Este algoritmo presentado en [4] fue una de las primeras soluciones para la búsqueda en espacios métricos en donde la función de distancia es discreta. Consiste en construir un árbol (*Burkhard-Keller Tree* o BKT) a partir de un nodo p elegido arbitrariamente. Para cada distancia $i > 0$ se define el subconjunto de los elementos que se encuentran a distancia i de la raíz p. Para cada uno de los subconjuntos no vacíos se crea un hijo de p, en donde recursivamente se construye un BKT. El proceso se repite hasta que quede un solo elemento en el subconjunto o hasta que queden menos de b elementos, que se almacenan en un bucket.

  Dada una consulta (q, r) y la función de distancia d del espacio métrico, sólo se examinan aquellas ramas en donde $d(p,q) - r \leq i \leq d(p,q) + r$, y se procede recursivamente en cada subárbol. Al llegar a una hoja, se realiza una búsqueda exhaustiva sobre todos los elementos contenidos en ella. La desigualdad triangular asegura que el algoritmo responde la consulta de manera correcta.

  Es fácil apreciar que las raíces p del BKT son los pivotes que utiliza el algoritmo para indexar el espacio métrico.

- Fixed-Queries Tree: En [5] se presenta el algoritmo *Fixed-Queries Tree* o FQT, que es una variante del algoritmo BKT. El FQT es un BKT en donde se ocupa un solo pivote para cada nivel del árbol, es decir el mismo pivote será la raíz de todos los subárboles de un mismo nivel.

  El algoritmo para responder consultas es exactamente el mismo que en el caso del BKT, pero en este caso sólo se compara q una vez por nivel. Por otro lado, los árboles FQT tienden a ser más altos que los BKT.

- Fixed-Height FQT: En [5] también se presenta el *Fixed-Height FQT* o FHQT, que es una variante del FQT, en donde todas las hojas se encuentran a la misma altura h, independientemente del tamaño de los buckets. El FHQT equivale exactamente a indexar el espacio con k pivotes fijos.

- Fixed Queries Array: En [6] se presenta el *Fixed Queries Array* o FQA. Este algoritmo representa en forma compacta, en un arreglo, un FHQT, lo que permite ahorrar memoria y por ende utilizar una mayor cantidad de pivotes.

- Vantage Point Tree: Este algoritmo [15] permite realizar consultas cuando la función de distancia es continua. El *Vantage Point Tree* o VPT consiste en construir un árbol binario, en donde la raíz es un elemento p elegido al azar. Luego se calcula la mediana del conjunto de todas las distancias, como $M = mediana\{d(p,u), u \in E\}$. Aquellos elementos u tal que $d(p,u) \leq M$ son insertados en el subárbol izquierdo, y los restantes son insertados en el subárbol derecho. Para resolver una consulta q con un radio de búsqueda r en el VPT, se calcula $dist = d(q, p)$. Si $dist - r \leq M$, entonces se entra en el subárbol izquierdo, y si $dist + r > M$ entonces se entra en el subárbol derecho. Nótese que es posible entrar en ambos subárboles a la vez. Todos los elementos que cumplan con la condición de la consulta son reportados.

- Multi Vantage Point Tree: El VPT puede ser extendido a árboles t-arios usando t-1 percentiles uniformes en vez de la mediana [7]. Este algoritmo recibe el nombre de *Multi Vantage Point Tree* o MVPT.

- Excluded Middle Vantage Point Forest: Otra extensión al VPT es el *Excluded Middle Vantage Point Forest* o VPF [8]. Si bien este algoritmo está diseñado para responder consultas por el elemento más cercano a q, se puede adaptar para realizar consultas por rango. El algoritmo consiste en excluir en cada nivel los elementos que se encuentren a distancias intermedias del pivote, y con esta parte excluida se construye un segundo árbol, con lo que se obtiene un bosque. Con esto es posible eliminar el backtracking al realizar búsquedas con un radio pequeño, pero hay que realizar la búsqueda en todos los árboles del bosque.

- Approximating Eliminating Search Algorithm: Un enfoque completamente distinto es el que ocupa el algoritmo *Approximating Eliminating Search Algorithm* o AESA [9]. El índice construido es simplemente una matriz en donde se precalculan las $\frac{n(n-1)}{2}$ distancias entre los elementos del espacio E. Para una consulta q con un radio de búsqueda r, se selecciona un elemento $p \in E$ al azar y se calcula $dist = d(p,q)$. Se excluyen todos los elementos $u \in E$ que no cumplan con la condición $dist - r \leq d(u, p) \leq dist + r$. Como todas las distancias $d(p,u)$ están precalculadas, solo

$d(p,q)$ debe ser calculada en el momento de la búsqueda. El proceso de elección de pivotes p se repite hasta que queden pocos elementos no excluidos, los cuales se comparan directamente con la consulta q.

El problema de este algoritmo es que tiene costo $O(n^2)$ en espacio y tiempo de construcción, lo cual es demasiado alto incluso para bases de datos pequeñas.

- Linear AESA: Una nueva versión del algoritmo AESA llamada *Linear AESA* o simplemente *LAESA* [10], propone utilizar una cantidad fija de pivotes, k. En este caso, el costo en espacio y tiempo de construcción es $O(kn)$. Aquellos elementos que no pueden ser excluidos después de considerar los k pivotes son comparados directamente con la consulta q (ver sección 2.4).

- Spaghettis: En [16] se presenta este algoritmo de búsqueda en espacios métricos basado en arreglos y que es una variante de LAESA. Propone reducir el tiempo de CPU extra necesario al realizar una consulta utilizando una estructura de datos en donde las distancias a los pivotes están ordenadas por separado, lo que permite utilizar una búsqueda binaria en el rango relevante (ecuación 2-6), pero necesita punteros extras para poder realizar la traza de un elemento a través de los diferentes órdenes por cada pivote.

Otra forma de indexar un espacio métrico se denomina ***clustering***, y consiste en particionar el espacio métrico en **clases**, donde cada partición se caracteriza por poseer un centro c_i y la clase respectiva se compone de los elementos del espacio en donde c_i es su centro más cercano. Al realizar una consulta se descartan la mayor cantidad de clases posibles, y se realiza una búsqueda exhaustiva dentro de las clases restantes.

Algunos algoritmos que utilizan este tipo de indexación para realizar búsquedas en espacios métricos son: *Generalized-Hyperplane Tree* [17], *Bisector Trees* [18], *Voronoi Tree* [19], *M-tree* [20] y *Spatial Approximation Tree* [21].

4. Dimensión intrínseca

Como se ha mencionado anteriormente, el rendimiento de los algoritmos de búsqueda en espacios métricos basados en pivotes empeora cuando la dimensión del espacio aumenta. En el caso de los espacios vectoriales, su dimensión representacional es el número de coordenadas de los vectores que representan al conjunto; sin embargo, un espacio vectorial de alta dimensionalidad representacional puede poseer **intrínsecamente** una dimensionalidad baja, por ejemplo, cuando el conjunto de puntos de un espacio vectorial de dimensión 20 se puede representar como un plano inmerso en dicho espacio, por lo que su dimensión intrínseca sería solo 2.

A continuación se presentará el concepto de dimensión intrínseca de un espacio métrico y la llamada maldición de la dimensionalidad.

4.1 Definición de dimensión intrínseca

La distribución de distancias de un espacio métrico se define como la probabilidad que dos elementos aleatorios del espacio se encuentren a una cierta distancia l [22]. Una aproximación a esta distribución es el **histograma de distancias**, el cual se construye a partir de un subconjunto del espacio métrico, calculando las distancias entre los elementos de dicho subconjunto.

En [11] se define la **dimensión intrínseca** de un espacio métrico como:

$$\rho = \frac{\mu^2}{2\sigma^2}$$

Ecuación 4-1

Los parámetros μ y σ^2 de la ecuación 4-1 son la media y la varianza, respectivamente, del histograma de distancias de los puntos que conforman dicho espacio métrico.

En el caso de los espacios vectoriales de dimensión k y usando la familia de funciones L_s, se obtiene que la dimensión intrínseca de un espacio vectorial, donde las coordenadas de los puntos fueron elegidas aleatoriamente, es de $\Theta(k)$, lo cual le da un sentido intuitivo a esta definición de dimensión intrínseca.

Para ilustrar lo anterior, se realizó el siguiente experimento: se generó un conjunto de $n = 10.000$ puntos con coordenadas aleatorias, pertenecientes a un espacio vectorial de dimensión k, $k = 2..20$. Se calculó la dimensión intrínseca de dichos conjuntos utilizando las funciones de distancia L_1, L_2 y L_∞.

El resultado del experimento se muestra en el gráfico 4-1. Se observa claramente que la dimensión intrínseca del espacio vectorial aleatorio aumenta linealmente con la dimensión representacional del espacio. También es posible observar que a medida que el **s** de la función de distancia crece, la constante asociada al $\Theta(k)$ también crece, pero está acotada por la constante asociada a L_∞. Las pendientes obtenidas de los datos se muestran en la tabla 4-1. Las pendientes asociadas a las demás distancias L_s se encuentran entre la pendiente asociada a L_2 y la pendiente asociada a L_∞.

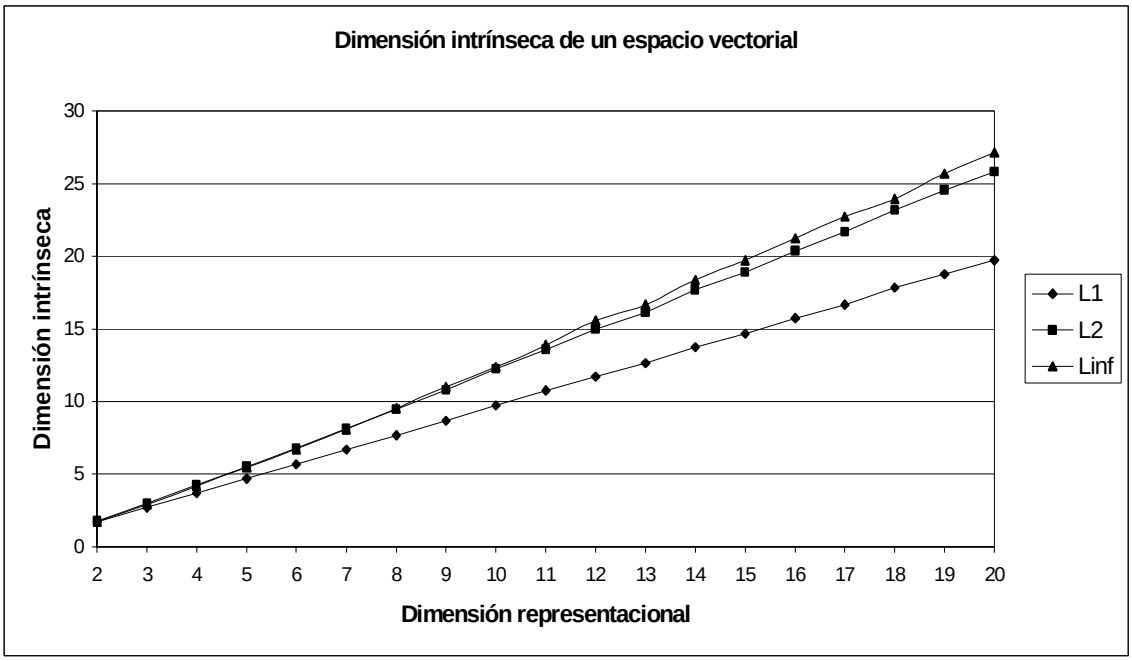

Gráfico 4-1: Dimensión de un espacio vectorial aleatorio vs su dimensión intrínseca

Función de distancia	Pendiente obtenida
L_1 (Manhattan)	1.00272688
L_2 (Euclidiana)	1.34565602
L_∞ (distancia del máximo)	1.42545452

Tabla 4-1: Pendientes de las rectas dimensión vs dimensión intrínseca

4.2 La maldición de la dimensionalidad

Cuando se realiza una consulta por rango, esto es, recuperar los elementos del espacio métrico que se encuentran a una distancia menor que un radio dado del elemento consultado, los algoritmos basados en pivotes buscan descartar la mayor cantidad de elementos del espacio métrico antes de realizar una búsqueda exhaustiva, es decir, generan una lista de elementos candidatos en la cual se verifica exhaustivamente la condición de búsqueda.

Considerando nuevamente el histograma de distancias de un espacio métrico (E, d), es fácil ver que según la condición de exclusión de elementos (ecuación 2-6), dada una consulta (q, r) y un conjunto de k pivotes p_i, se pueden descartar todos los elementos $x \in E$ que no cumplan para algún $i \in 1..k$ la condición de la ecuación 4-2:

$$d(p_i, x) \notin [d(p_i, q) - r, d(p_i, q) + r]$$

Ecuación 4-2

Mientras mayor sea la dimensión intrínseca del espacio métrico, la media del histograma aumenta y/o su varianza disminuye.

Cuando la varianza del histograma disminuye a medida que aumenta la dimensión del espacio, la cantidad de elementos que se encuentran dentro del rango $[d(p,q)-r, d(p,q)+r]$ también aumenta, es decir, cada vez son menos los elementos que se pueden descartar. Alternativamente, si la media crece entonces r crece con la dimensión del espacio si se desea obtener un porcentaje fijo de elementos del conjunto. En espacios de alta dimensionalidad, prácticamente todos los elementos se transforman en candidatos a la verificación exhaustiva (complejidad externa), por lo que deben ser comparados directamente con la consulta. A esto se le denomina maldición de la dimensionalidad, y es independiente de la naturaleza del espacio métrico al cual pertenecen los elementos.

Las figuras 4-1 y 4-2 nos muestran de manera intuitiva la maldición de la dimensionalidad. El histograma de distancias de la figura 4-1 representa a un espacio métrico de dimensión baja, y el histograma de distancias de la figura 4-2 representa a un espacio métrico de dimensión alta. Se puede apreciar que en el caso del espacio con dimensión alta casi ningún elemento puede excluirse de la lista de candidatos (el área gris), por lo que debe realizarse prácticamente una búsqueda exhaustiva para responder la consulta. Para poder descartar elementos es necesario utilizar más pivotes, pero esto aumenta la complejidad interna de la consulta.

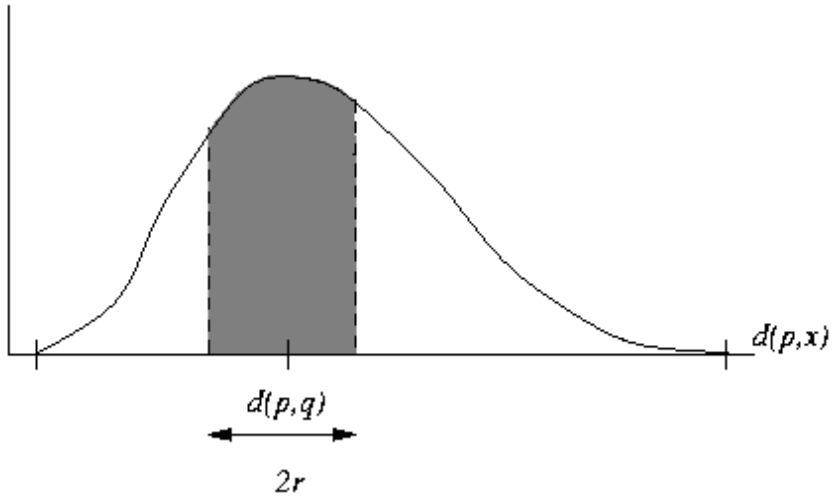

Figura 4-1: Histograma de distancias de un espacio métrico de dimensión baja

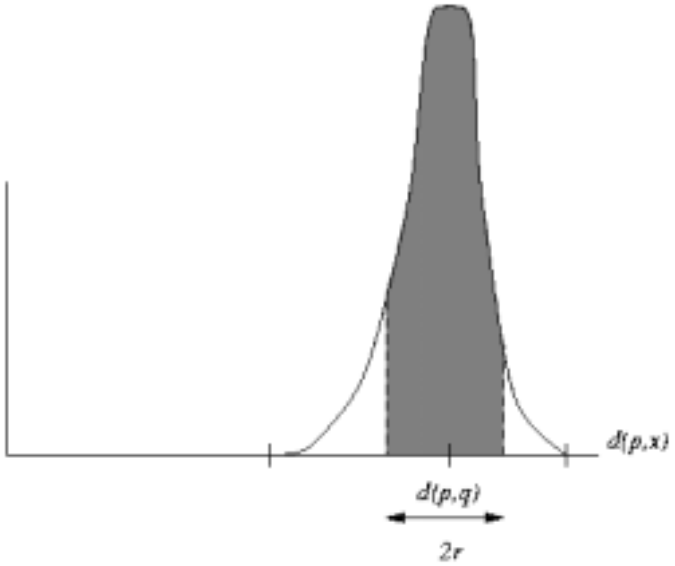

Figura 4-2: Histograma de distancias de un espacio métrico de dimensión alta.

En [11] se demuestra que cualquier algoritmo basado en pivotes, en donde los pivotes se eligen al azar, tiene una cota inferior de $\frac{r^2}{2\sigma^2}\log(n)$ evaluaciones de distancia en promedio para una consulta por rango. Para recuperar una fracción f del total de elementos del conjunto, por Chebyschev (ver sección 2.5) se tiene que $r = \mu - \frac{\sigma}{\sqrt{f}}$, por lo que esta cota inferior se puede expresar en términos de la dimensión intrínseca del espacio métrico como [23]:

$$Costo \geq \left(\sqrt{\rho} - \frac{1}{\sqrt{2f}} \right)^2 \log(n)$$

Ecuación 4-3

Esto se logra usando el número óptimo de pivotes, que también crece como $\Theta(\rho \log(n))$. La maldición de la dimensionalidad indica lo difícil que se torna el problema de la búsqueda aproximada en espacios métricos cuando la dimensión del espacio es alta.

5. Técnicas de selección de pivotes

5.1 Criterios de eficiencia

Lo que se busca es minimizar el número de evaluaciones de distancia al realizar una consulta por rango. Para lograr esto, el índice que se construye a partir de los pivotes escogidos debe excluir la mayor cantidad de elementos posibles antes de realizar búsqueda exhaustiva dentro de la lista de elementos candidatos (elementos no excluidos). Por lo tanto, un buen conjunto de pivotes debe generar una lista de elementos candidatos lo más pequeña posible.

Sea (E,d) un espacio métrico. Un conjunto de pivotes $\{p_1,..,p_k\}, p_i \in E$ definen un espacio P de tuplas de distancias entre pivotes y elementos del conjunto. El mapeo de un elemento $x \in E$ a P, que se denotará $[x]$, está dado por la ecuación 5-1:

$$[x] = (d(x,p_1), d(x,p_2),...d(x,p_k)), x \in E, [x] \in P$$

Ecuación 5-1

Definiendo la métrica $D = D_{\{p_1,..,p_k\}}$ del espacio P como:

$$D([q],[x]) = \max_{i=1}^{k} |d(q,p_i) - d(x,p_i)|$$

Ecuación 5-2

Se obtiene el espacio métrico (P,D), que resulta ser $(\Re^k, L_\infty)$. Dada una consulta por rango (q,r) es fácil ver en este nuevo espacio métrico que, según la condición de la ecuación 2-6, no se pueden excluir aquellos elementos $x \in E$ que cumplan:

$$D_{\{p_1,..,p_k\}}([q],[x]) \le r$$

Ecuación 5-3

Para lograr que la cantidad de elementos no excluidos sea pequeña se debe maximizar la probabilidad que $D_{\{p_1,...,p_k\}}([q],[x]) > r$. Una forma de lograr esto es maximizar la media de la distribución de distancias en P, la cual se denotará μ_p.

Por lo tanto, la manera de comparar dos conjuntos de pivotes será comparando cuál de ellos hace que μ_p sea mayor. Esta es una alternativa para comparar conjuntos de pivotes, pero también existen otras, como por ejemplo maximizar la dimensión intrínseca ρ_p del espacio métrico (P, D), que correspondería a maximizar μ_p y a minimizar la varianza de P, σ_p. Se eligió μ_p antes que ρ_p porque los resultados experimentales fueron mejores al maximizar μ_p.

5.2 Estimación de μ_p

La estimación de μ_p se realiza de la siguiente forma:

- Se eligen A pares de elementos $\{(a_1, a'_1), (a_2, a'_2), ..., (a_A, a'_A)\}$ del conjunto E, distintos entre sí.
- Se mapean los A pares de elementos al espacio P y se calcula la distancia D entre cada par de elementos, con lo que se obtiene el conjunto de distancias $\{D_1, D_2, ..., D_A\}$.
- Una vez obtenidas las A distancias , se estima el valor de μ_p como:

$$\mu_p = \frac{\sum_{i=1}^{A} D_i}{A}$$

Ecuación 5-4

De las ecuaciones 5-1 y 5-2 se deduce que, dado un par de elementos (a, a') el costo de calcular $D([a],[a'])$ es de $2k$ evaluaciones de la función d.

Si se utilizan A pares de elementos para estimar el valor de μ_p, es fácil ver que el costo total de la estimación es de $2kA$ evaluaciones de la función d.

5.3 Métodos de selección de pivotes

A continuación se describen los métodos de selección de pivotes utilizados, y se describe el costo de cada uno de ellos en función del número de evaluaciones de la distancia d.

5.3.1 Selección aleatoria

Es la base del resto de la investigación. Se realizarán experimentos para calcular el número óptimo de pivotes en los siguientes casos:

- Número de elementos del conjunto fijo, dimensión del espacio métrico variable.
- Dimensión del espacio métrico fija, número de elementos del conjunto variable.

Dentro de cada uno de estos casos se variará la cantidad de pivotes del índice a construir.

Los resultados obtenidos con estos experimentos servirán como punto de referencia para ver qué tanto mejora el algoritmo basado en pivotes, utilizando alguna técnica de selección de pivotes, con respecto de elegir los pivotes al azar.

Naturalmente, el costo de optimizar los pivotes en este método es 0.

En el apéndice A.1 se describe en pseudocódigo la programación de este método de selección.

5.3.2 Selección de N conjuntos aleatorios

Se eligen N conjuntos de k pivotes al azar, y se calcula la media μ_p de la distribución de distancias de P para cada uno de ellos, utilizando A pares de elementos escogidos al azar del espacio métrico E. El conjunto elegido es aquel que maximiza μ_p.

En el apéndice A.2 se describe en pseudocódigo la programación de este método de selección.

Costo del algoritmo

Dado que el algoritmo realiza N estimaciones de μ_p, el trabajo total realizado es:

$Trabajo = 2kAN$ evaluaciones de la función d.

5.3.3 Selección incremental

Se elige un pivote p_1 utilizando A pares de elementos del espacio E mapeados a P, tal que ese pivote maximice μ_p. Luego se elige un segundo pivote p_2, tal que $\{p_1, p_2\}$ maximicen μ_p pero p_1 ya se considera fijo. Luego se elige un tercer pivote p_3, tal que $\{p_1, p_2, p_3\}$ maximicen μ_p considerando p_1 y p_2 fijos. El proceso se repite para elegir los pivotes restantes, y se termina cuando se han elegido k pivotes.

En cada iteración, el pivote se elige dentro de una muestra de tamaño X del espacio E, puesto que buscar aquel elemento que maximice μ_p dentro de todo el conjunto E resultaría muy costoso.

En el apéndice A.3 se describe en pseudocódigo la programación de este método de selección.

Costo del algoritmo

Si bien en cada iteración se estima μ_p con los i pivotes elegidos hasta el momento, no es necesario rehacer el cálculo completo si se almacena $D_{\{p_1,\ldots,p_{i-1}\}}([a_r],[a'_r]), \forall r \in 1..A$, es decir, para cada valor de $r = 1..A$ el valor máximo de $\left|d(a'_r, p_j) - d(a_r, p_j)\right|, j = 1..i-1$. En este caso, sólo se calcula la distancia con respecto a los pivotes candidatos, $\left|d(a'_r, p_{cand}) - d(a_r, p_{cand})\right|, r = 1..A$, y se toma el valor máximo entre las dos distancias para el cálculo de $D_{\{p_1,\ldots,p_i\}}$. Por lo tanto, si la muestra de donde se toman los pivotes candidatos es de tamaño X, en cada iteración se realizan $2AX$ evaluaciones de la función d. Si se eligen k pivotes, el trabajo total realizado por el algoritmo es:

$Trabajo = 2kAX$ evaluaciones de la función d.

5.3.4 Selección de óptimo local

Se elige un conjunto de k pivotes al azar. Se calcula la matriz $M(r, j) = \left| d(a_r, p_j) - d(a'_r, p_j) \right|, r = 1..A$. Es fácil ver que $D([a_r],[a'_r]) = \max_{j=1}^{k}(M(r, j))$ para un r fijo, y a partir de estos valores se puede estimar μ_p. Además, se deben almacenar por cada fila de la matriz M los índices de los pivotes en donde esta el máximo valor, que se denotará r_{max}, y segundo máximo valor, que se denotará r_{max2}. Se evalúa la contribución de cada pivote p_j como la suma sobre las A filas de cuánto aumenta $D([a_r],[a'_r])$ gracias a p_j, esto es:

- $M(r, r_{max}) - M(r, r_{max2})$ si $j = r_{max}$ en dicha fila.
- 0 en caso contrario.

Se elige el peor pivote del grupo, la víctima, como aquel que tiene la menor contribución dentro de los k pivotes, y se intenta cambiarlo por algún nuevo pivote candidato, elegido entre una muestra de X elementos pertenecientes al espacio métrico E. Se calcula la contribución de los X pivotes candidatos, y se elige aquel cuya contribución sea mayor. Si la contribución del pivote elegido es mayor que la contribución de la víctima, se saca la víctima del conjunto de pivotes y el pivote candidato pasa a formar parte del conjunto de pivotes. El proceso se repite N veces.

En el apéndice A.4 se describe en pseudocódigo la programación de este método de selección.

Costo del algoritmo

El costo de construcción de la matriz M es de $2Ak$ evaluaciones de la función d. El costo de calcular la víctima es 0, puesto que toda la información para calcular la contribución de cada pivote se encuentra en la matriz M. El costo de calcular la contribución de los X pivotes candidatos es de $2AX$ evaluaciones de distancia, y el proceso de repite N' veces, por lo que el costo total del algoritmo es de:

$$Trabajo = 2A(k + N'X) \text{ evaluaciones de la función } d.$$

Si se considera $kN = k + N'X$, es fácil ver que debe cumplirse $N'X = k(N-1)$, de esta forma el costo total del algoritmo es:

$$Trabajo = 2AkN \text{ evaluaciones de la función } d.$$

Nótese que es posible intercambiar X por N' manteniendo el mismo costo del algoritmo.

5.4 Comparación de costos entre los distintos métodos de selección

Haciendo un resumen de los costos de los métodos de selección, se tiene que:

- Selección N grupos al azar = $2kAN$ evaluaciones de distancia.
- Selección incremental = $2kAX$ evaluaciones de distancia.
- Selección de óptimo local = $2AkN$ evaluaciones de distancia.

Por lo tanto, a igual costo se tiene que $N = X$.

Sin embargo, el método de selección incremental posee las siguientes ventajas con respecto a los otros métodos de selección de pivotes:

- Al ser un método incremental se pueden agregar nuevos pivotes en cualquier momento, sin tener que realizar el cálculo desde cero, lo que no ocurre con ninguno de los otros métodos.
- Se puede determinar exactamente cuando parar para elegir el número óptimo de pivotes: basta iterar hasta que la complejidad total del algoritmo no mejore más. En cambio, con los otros métodos de selección hay que realizar el cálculo completo para distinto número de pivotes si se desea encontrar el número óptimo.
- Este método es fácilmente adaptable para utilizarlo con algoritmos como FHQT y similares (ver sección 3). Para el caso particular del FHQT se procede de la siguiente manera: se elige la raíz del árbol como aquel elemento que maximice D por si solo, con lo que se obtienen "bolsas" de elementos, cada una de ellas a distancia i de la raíz; recursivamente se elige el mejor pivote para el siguiente nivel tomando en consideración los pivotes previamente elegidos.

Para comparar los distintos métodos de selección se utilizará el algoritmo básico de búsqueda usando pivotes (ver sección 2.4), el cual es una variante del algoritmo LAESA. Por lo tanto, los resultados

obtenidos en los experimentos sólo serán directamente aplicables a los algoritmos que utilizan k pivotes fijos (FHQT, FQA, LAESA, Spaghetti), pero posiblemente se pueden extender a otros algoritmos que ocupan pivotes.

6. Resultados experimentales

6.1 Descripción de los experimentos

Los experimentos descritos en este capítulo abarcan los siguientes tópicos:

- Estudio de la incidencia del parámetro N en el rendimiento de los distintos métodos de selección.
- Estudio de la incidencia del parámetro A en el rendimiento de los distintas métodos de selección.
- Comparación del rendimiento de los distintos métodos de selección.
- Comparación del mejor método de selección con respecto a elegir los pivotes aleatoriamente, al variar el número de elementos del conjunto de datos, al variar la dimensión representacional del conjunto de elementos, al utilizar el número óptimo de pivotes y al disponer solamente de una cantidad fija de pivotes.
- Estudio de las características de un buen conjunto de pivotes.

Con respecto a los tres primeros puntos, se mostrarán los resultados de experimentos en donde se fija la cantidad de elementos del conjunto y su dimensión representacional, y se hace variar el número de pivotes que se utiliza.

Los conjuntos de elementos son espacios vectoriales, un caso particular de espacios métricos, en donde su dimensión varía entre 2 y 24 dimensiones y contienen entre 10.000 y 100.000 elementos uniformemente distribuidos en el cubo unitario, es decir, cada coordenada x_i del espacio vectorial tiene su valor entre $[0,1)$. Para calcular la complejidad total de cada método utilizando k pivotes, se realizaron 1.000 consultas al azar con un radio de búsqueda calculado para retornar, aproximadamente, el 0.01% del total de elementos del conjunto. Para responder cada consulta solo se ocupó la función de distancia L_2 entre dos pares de elementos, y no se hizo uso de la información que proveen las coordenadas de los puntos en el espacio vectorial; de este modo se trató el espacio vectorial como un espacio métrico en el que se puede controlar la dimensión real. Toda la información con respecto a los parámetros de cada experimento realizado se describirán en los títulos de los gráficos presentados.

Para comparar los métodos de selección de manera equitativa, se le permitirá a cada método la misma cantidad de trabajo al optimizar los pivotes, es decir, podrán realizar la misma cantidad de evaluaciones de distancia al momento de construir el índice, lo que implica fijar los parámetros N y A. Por lo tanto, los métodos de selección de pivotes a comparar son:

- **Azar:** método aleatorio de selección de pivotes.
- **Sele**: método de selección de N grupos aleatorios (ver sección 5.3.1).
- **Incr**: método incremental de selección, con $X = N$ (ver sección 5.3.2).
- Método de selección de óptimo local. En este caso, dado que se tiene $N' X = k(N-1)$ (ver sección 5.3.4), se probarán las siguientes alternativas:
  - **OptA**: $N' = k$ y $X = N-1$, es decir se eligen k víctimas y la muestra de los posibles reemplazantes es de tamaño $N-1$. A este método se le denominará **óptimo local A.**
  - **OptB**: $N' = N-1$ y $X = k$, es decir se eligen $N-1$ víctimas y la muestra de los posibles reemplazantes es de tamaño k. A este método se le denominará **óptimo local B**.

No se utilizó otra combinación posible para el método de óptimo local, cuando $N' = X = \sqrt{k(N-1)}$, pues los resultados obtenidos muestran que no funciona mejor que las otras combinaciones y no aporta más información sobre el rendimiento del método.

Los experimentos fueron programados y ejecutados en un PC con un procesador Intel Pentium III de 450 Mhz y 128 Mb de memoria física. Los programas fueron desarrollados en lenguaje C y compilados con GNU C Compiler (versión 2.91.66) bajo ambiente Linux (kernel 2.2.14-5.0).

6.2 Resultados obtenidos

6.2.1 Parámetro N

Si bien el parámetro N del costo de los métodos de selección de pivotes no significa lo mismo para todos los métodos, el gráfico 6.1 muestra que en realidad basta con un N pequeño para obtener un buen rendimiento del método, exceptuando el método de óptimo local B. Para la realización de este experimento se fijaron tanto el número de pivotes como el valor del parámetro A.

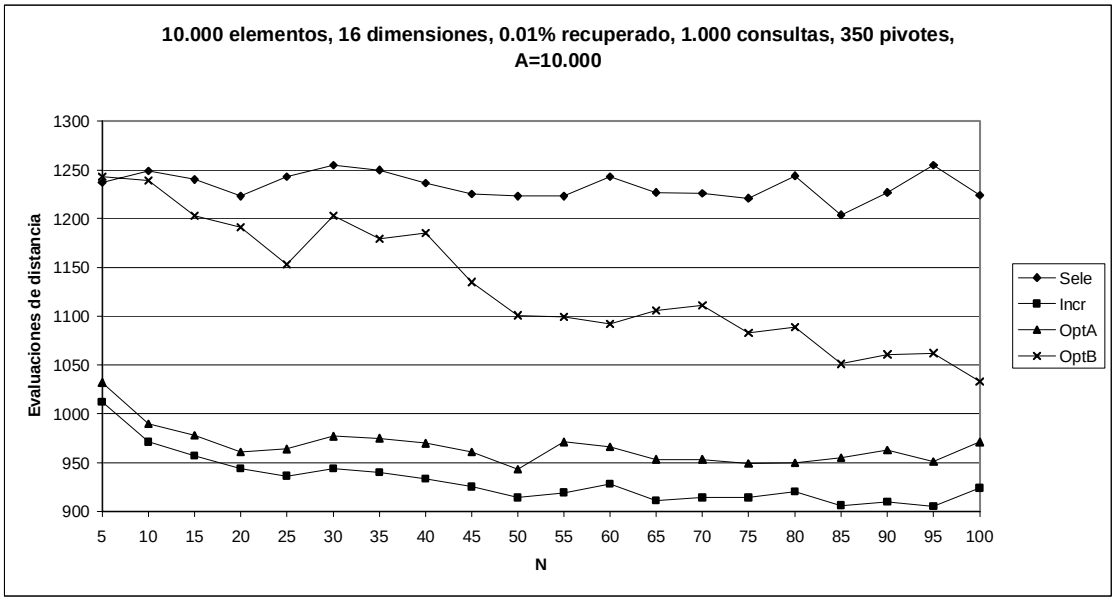

Gráfico 6-1: Rendimiento de las técnicas de selección al variar N entre 5 y 100

Se observa del gráfico que a partir de un N bastante pequeño ($N = 20$) el aumento del rendimiento de las técnicas de selección es bajo, excepto con el método óptimo local B, lo cual es lógico puesto que en ese caso el parámetro N es proporcional al número de veces que se elegirá una víctima para ser reemplazada por el mejor pivote dentro de una muestra de elementos del conjunto. Al aumentar entre $N = 5$ y $N = 100$ la técnica de selección incremental mejora en un 8.7%

Se torna más evidente lo poco que mejoran las técnicas de selección de pivotes cuando se hace aumentar N en valores mayores, como lo indica el gráfico 6-2. En este caso, la mejora en el número de evaluaciones de distancia es de un 6.28% entre $N = 50$ y $N = 1000$ para la técnica de selección incremental.

Se concluye de ambos gráficos que basta con un N pequeño para que las técnicas de selección elijan buenos conjuntos de pivotes.

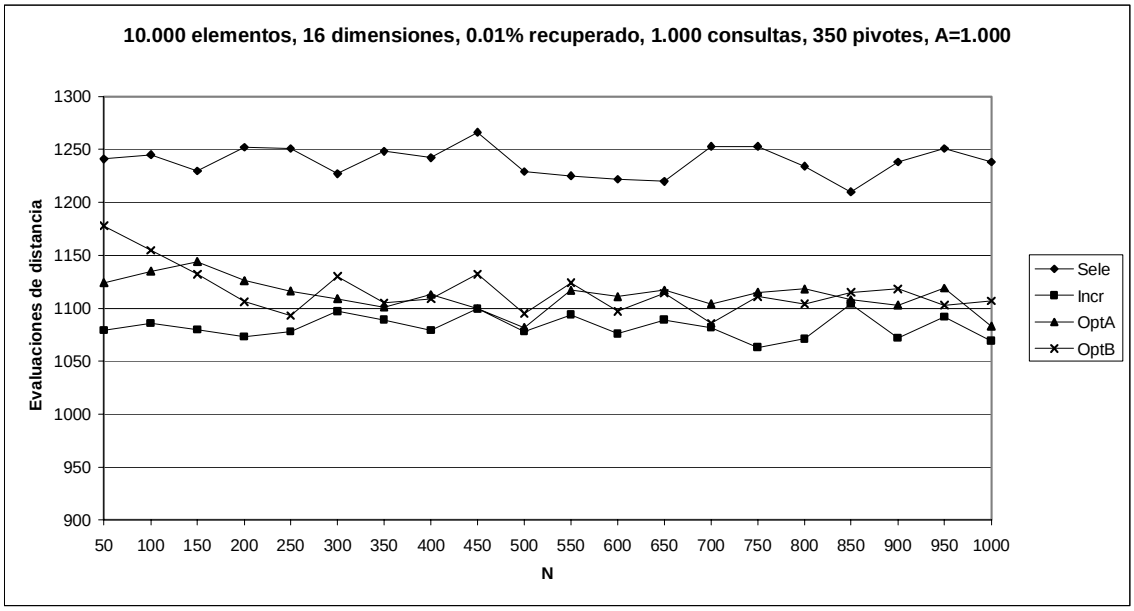

Gráfico 6-2: Rendimiento de las técnicas de selección al variar N entre 50 y 1.000

6.2.2 Parámetro A

Este parámetro sí tiene el mismo significado en todos los métodos de selección, y representa el número de pares de elementos, escogidos al azar, que se utilizarán para la estimación de μ_p. El gráfico 6.3 muestra el rendimientos de los distintos métodos de selección al hacer variar A, teniendo N y el número de pivotes fijos.

En este caso sí se nota una mejora continua en el rendimiento de las técnicas de selección al aumentar el valor de A. Dicha mejora es de un 16.25% para el método incremental y de un 19.90% para el método de selección de óptimo local A. El gráfico 6-4 muestra que el resultado obtenido con el experimento anterior se mantiene si se aumenta el valor del parámetro N.

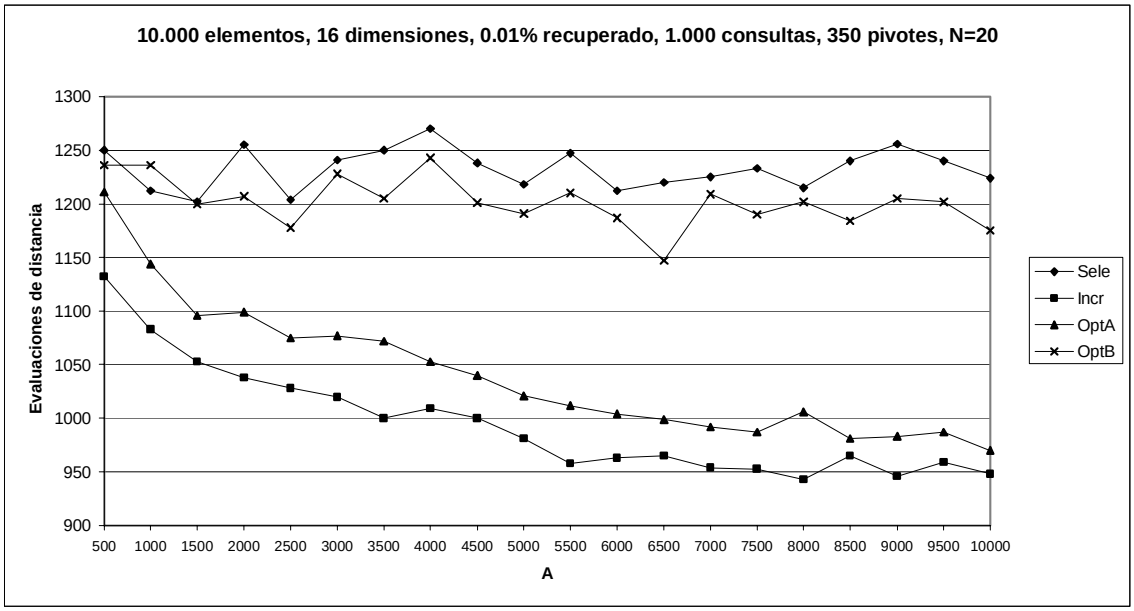

Gráfico 6-3: Rendimiento de las técnicas de selección al variar el parámetro A, $N = 20$

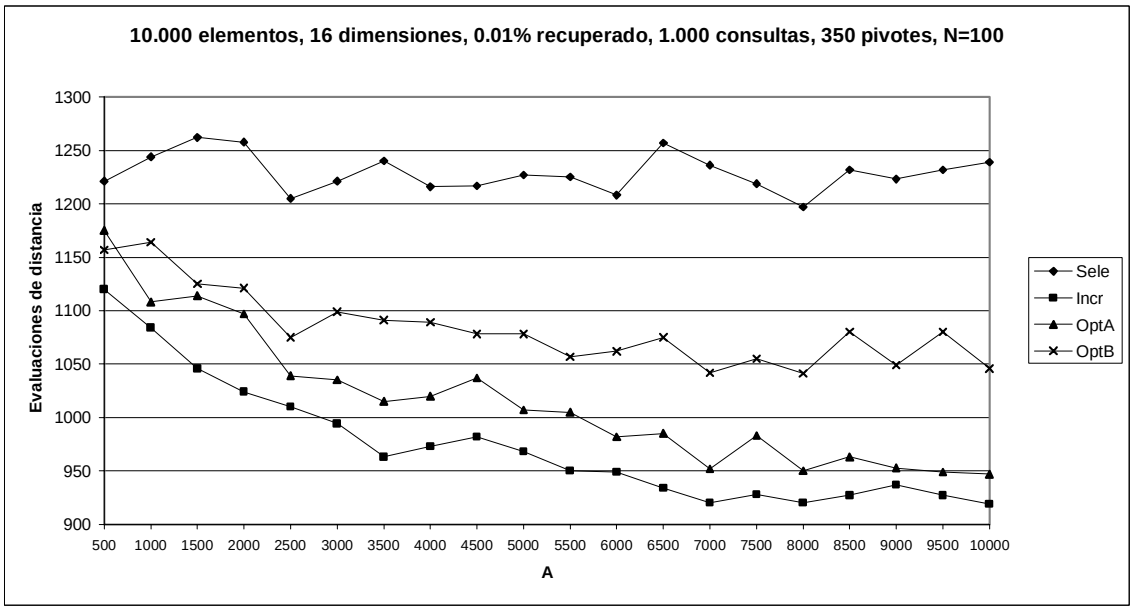

Gráfico 6-4: Rendimiento de las técnicas de selección al variar el parámetro A, $N = 100$.

La explicación de por qué al aumentar el valor de A el rendimiento de las técnicas de selección mejoran más que al aumentar el valor de N, exceptuando el método de óptimo local B por las razones previamente descritas, proviene de lo siguiente: los métodos de selección de pivotes maximizan la media de la distribución de distancias de los elementos del espacio métrico mapeados al espacio de pivotes.

Mientras más pares de elementos se utilicen para estimar la media, es decir, mientras mayor sea A, se obtiene un mayor poder discriminativo entre dos conjuntos de pivotes.

Supóngase que se tienen pocos pares de elementos para estimar μ_p, y que se quiere agregar un pivote a un conjunto previamente elegido, al estilo del método incremental de selección. Dado que los pares de elementos son pocos, es posible que la distancia D con respecto al nuevo pivote nunca sea mayor a ninguna de las calculadas con los pivotes elegidos previamente, por lo que el nuevo pivote, aparentemente, no agrega nueva información con respecto al conjunto que ya había sido elegido previamente. A medida que se aumenta el número de pares de elementos con el que se estima μ_p, la posibilidad que esto ocurra disminuye: basta con que una sola distancia D de todos los pares de elementos cambie de valor para que el valor de la estimación de μ_p varíe, y por lo tanto es posible distinguir cuál pivote es el que aumenta más la media. Nótese que si D varía para un solo par de elementos al agregar un nuevo pivote, el valor de μ_p aumenta puesto que D es la métrica del espacio $(\Re^k, L_\infty)$ (ecuación 5-2).

Se concluye de lo anterior que es necesario tener un valor de A alto para que los métodos de selección de pivotes tengan un buen rendimiento.

Dado que el trabajo total al optimizar los pivotes es proporcional a NA, es útil saber que no vale la pena hacer crecer N en detrimento de A, y que si es posible aumentar la cantidad de trabajo a realizar en la optimización de pivotes entonces es preferible aumentar el valor de A antes que el valor de N.

6.2.3 Comparación de las técnicas de selección de pivotes

El gráfico 6-5 muestra el resultado de comparar las distintas técnicas de selección en un espacio vectorial de 8 dimensiones, donde los elementos del espacio están uniformemente distribuidos en el cubo unitario.

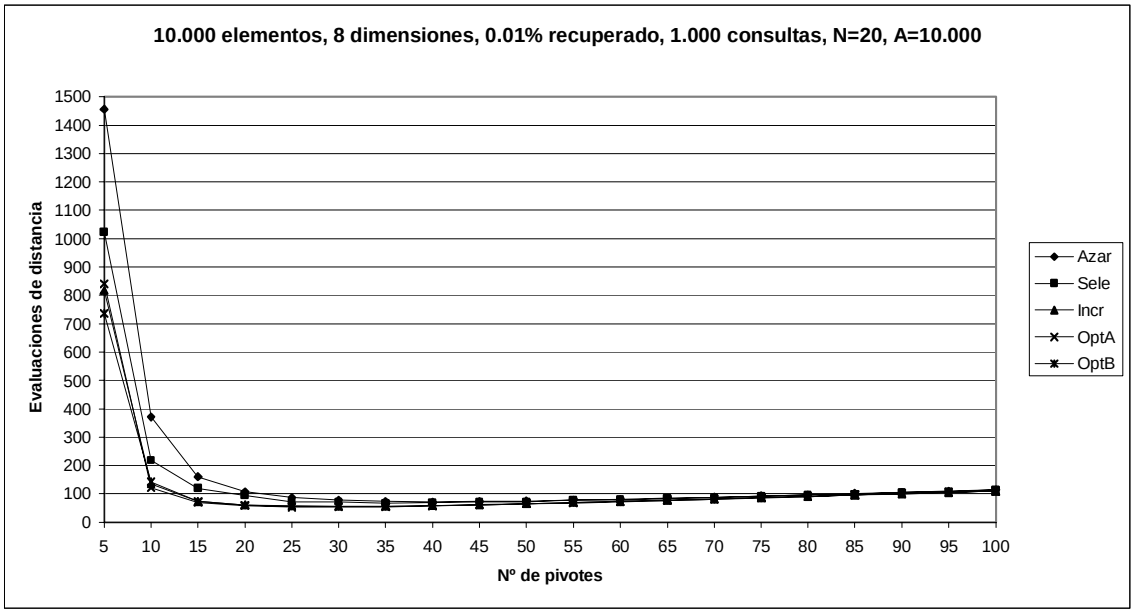

Gráfico 6-5: Comparación de técnicas de selección en espacio de 8 dimensiones, 10.000 elementos

Se observa que los métodos de selección incremental y de óptimo local A son los que obtienen un mejor rendimiento, en donde el número óptimo de pivotes disminuye de 40 (método de selección aleatorio) a 20, y disminuye el número de evaluaciones de distancia en el óptimo de 72 e. d. (evaluaciones de distancia) a 59 e. d. Si bien el número total de evaluaciones de distancia no mejora notablemente (18.06%), **el número de pivotes que se necesita para realizar el mismo trabajo que en el óptimo del método aleatorio disminuye drásticamente de 40 a 15 pivotes (62.5%)**, es decir se requiere solo un **37.5%** de la memoria que necesita el índice construido por el método aleatorio para realizar el mismo trabajo, en promedio, por consulta realizada.

El gráfico 6-6 muestra el mismo experimento anterior pero ahora con un conjunto de 100.000 elementos.

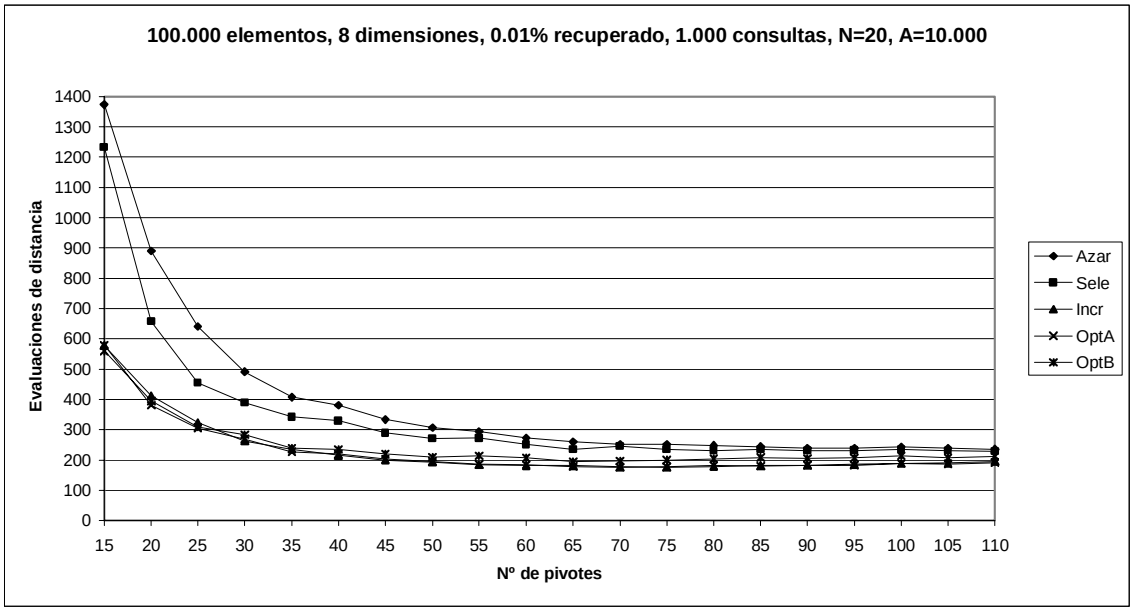

Gráfico 6-6: Comparación de técnicas de selección en espacio de 8 dimensiones, 100.000 elementos

En este gráfico ya se vislumbra que el método de selección de N grupos aleatorios no es una buena idea, lo cual tiene un explicación simple: son demasiados los grupos de pivotes que se pueden escoger desde el conjunto de elementos, de hecho si el conjunto tiene n elementos y se quieren escoger conjuntos de k pivotes, el número total de elecciones posibles es de $\binom{n}{k}$.

También en el gráfico 6-6 se observa que el número de e. d. en el óptimo del método aleatorio disminuye con respecto al número de e. d. en el óptimo del mejor método de selección: varía de 236 e. d. a 176 e. d. (25.42%), pero el número de pivotes que se necesitan para realizar el mismo trabajo que en el óptimo usando el método aleatorio disminuye drásticamente: de 110 pivotes a 35 pivotes (68.18%). Los mejores métodos de selección en este gráfico son el método incremental y el de óptimo local A.

El gráfico 6-7 compara el resultado del experimento para un espacio vectorial de 10.000 elementos y 16 dimensiones.

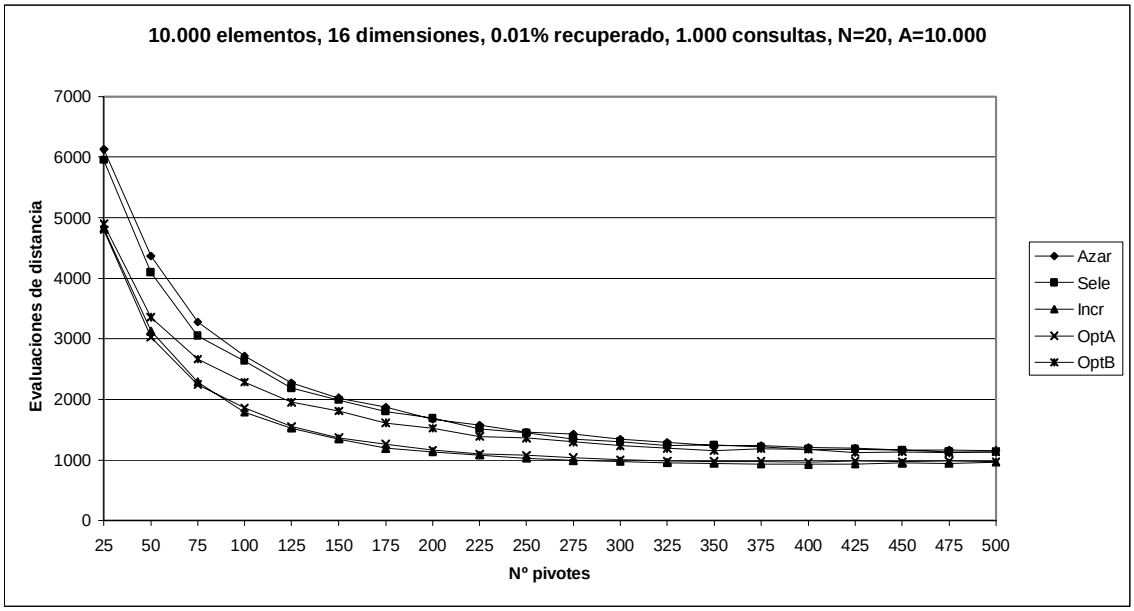

Gráfico 6-7: Comparación de técnicas de selección en espacio de 16 dimensiones, 10.000 elementos

En este caso, el óptimo mejora de 500 a 400 pivotes (método aleatorio y método incremental, respectivamente), y el número de e. d. en el óptimo mejora de 1162 a 960 e. d. (17.38%). También se observa que para realizar el mismo trabajo que en el óptimo con el método aleatorio se requieren muchos menos pivotes utilizando el método incremental de selección; con el método aleatorio se requieren 500 pivotes, mientras que con el método incremental se requieren solo 200 pivotes (60% menos de utilización de memoria). Los mejores métodos siguen siendo el incremental y el de óptimo local A. El método de selección de N grupos se comporta prácticamente igual que al elegir los pivotes aleatoriamente.

También se nota que el método de óptimo local B no es una buena idea, puesto que como el valor del parámetro N es pequeño, se reemplazan pocas víctimas por elementos que cumplen las condiciones de ser buenos pivotes. Sin embargo, es interesante notar que aún cambiando pocos pivotes con respecto a una elección inicial aleatoria, el método mejora el rendimiento del algoritmo de búsqueda cuando la cantidad de pivotes es pequeña.

El gráfico 6-8 muestra los resultados del experimento para un espacio de 10.000 elementos y 24 dimensiones, espacio que cae dentro de la categoría de los “intratables”.

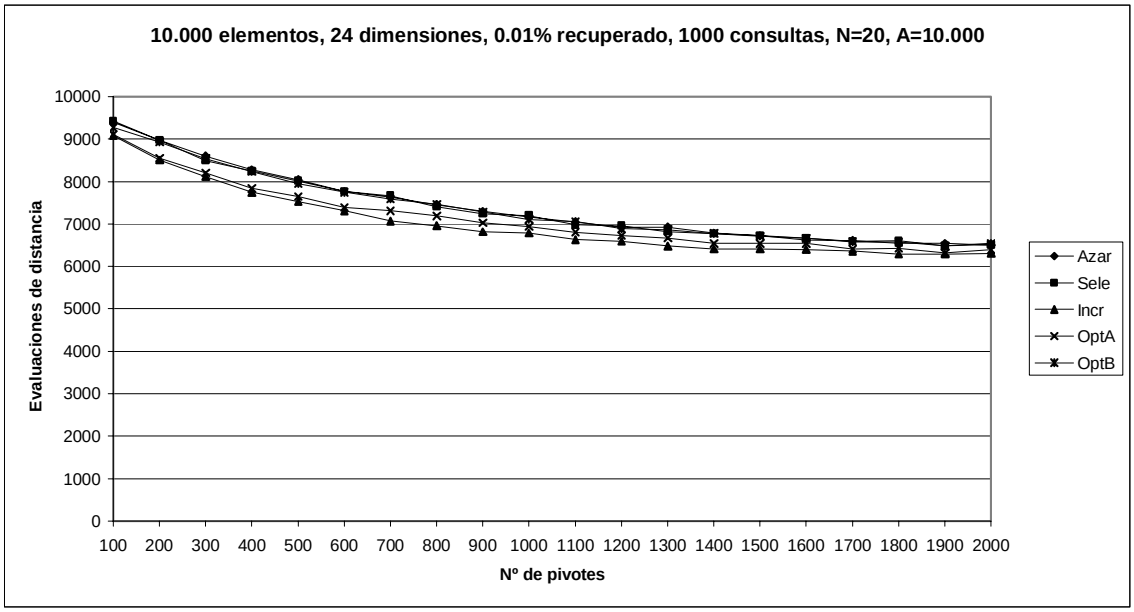

Gráfico 6-8: Comparación de técnicas de selección en espacio de 24 dimensiones, 10.000 elementos

En este punto la maldición de la dimensionalidad hace que la mejora en el número de e. d. sea muy baja en el óptimo. Nótese que en este caso no se alcanzó el óptimo con el método aleatorio ni siquiera utilizando el número máximo de pivotes, y el número de e.d. con este método utilizando 2.000 pivotes es de 6.501 e. d., mientras que con el método incremental se realizaron un óptimo de 6.287 e.d. con 1.800 pivotes, es decir con este método sí se alcanzó a llegar al número óptimo de pivotes. A pesar que la mejora en e. d. es mínima (3.29%), nuevamente se necesitan pocos pivotes para realizar el mismo trabajo que con el método aleatorio usando el máximo de pivotes: con el método incremental se necesitan 1.300 pivotes, lo que implica un ahorro de memoria del 35%.

De los resultados anteriores se concluye que el **método de selección incremental de pivotes** es el que funciona mejor en la práctica, y será el método de selección a utilizar en la realización del resto de los experimentos, ya que sólo es alcanzado por el método de óptimo local A pero tiene todas las ventajas adicionales mencionadas en la sección 5.4.

6.3 Comparación del método de selección incremental con la selección aleatoria de pivotes

El gráfico 6-9 muestra una comparación del rendimiento entre el método de selección incremental de pivotes y la elección aleatoria de pivotes al aumentar la dimensión del espacio vectorial, manteniendo fija

la cantidad de elementos del espacio. Están representados en el gráfico tanto la complejidad interna de ambos métodos de selección, que corresponde al número óptimo de pivotes en cada dimensión y es proporcional al espacio que ocupa el índice que se construye con dichos pivotes, y la complejidad total de ambos métodos. Es fácil calcular la complejidad externa, dado que es la complejidad total menos la complejidad interna.

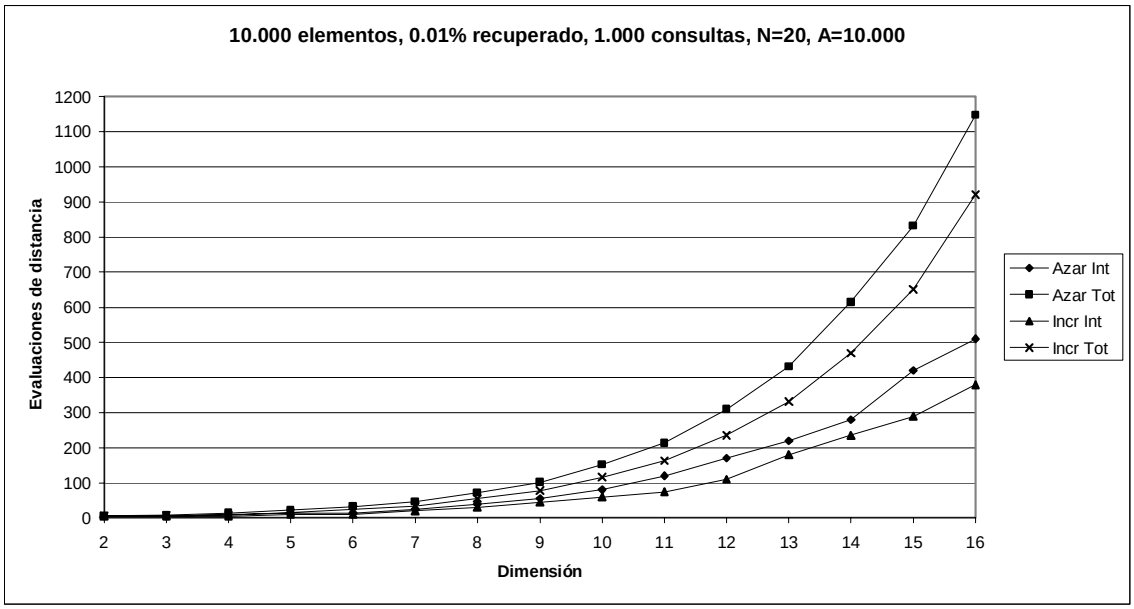

Gráfico 6-9: Número óptimo de pivotes según dimensión del espacio usando métodos de selección aleatoria e incremental

El gráfico 6-10 muestra los resultados del mismo experimento pero sólo hasta un espacio vectorial de dimensión 9, ya que es difícil apreciar este segmento del gráfico en la ilustración anterior. Se aprecia de ambos gráficos que en todo el rango estudiado se alcanza la complejidad total óptima con menos pivotes al utilizar el método de selección incremental de pivotes, y dicha complejidad total es menor a la que se obtiene con el número óptimo de pivotes al escoger éstos aleatoriamente.

La cota inferior mostrada en la ecuación 4.3 predice un comportamiento lineal para la curva del método aleatorio en función de la dimensión, sin embargo los gráficos muestran que en la práctica el costo de búsqueda aumenta más rápido que linealmente con la dimensión, y que usando el método de selección de pivotes este aumento es menos pronunciado que con el método aleatorio. Utilizando el método de los mínimos cuadrados se obtiene la siguiente aproximación para las curvas de complejidad total del gráfico 6-9:

$y = cx^{\alpha}$:

- Método de selección aleatorio:
  - $\alpha = 2.65044518 \pm 0.16914653$ (Error porcentual: 6.38%).
  - $c = 0.41992835 \pm 0.18227380$ (Error porcentual: 43.41%).
- Método de selección incremental:
  - $\alpha = 2.60074657 \pm 0.20152522$ (Error porcentual: 7.75%).
  - $c = 0.35760169 \pm 0.19185993$ (Error porcentual: 53.65%).

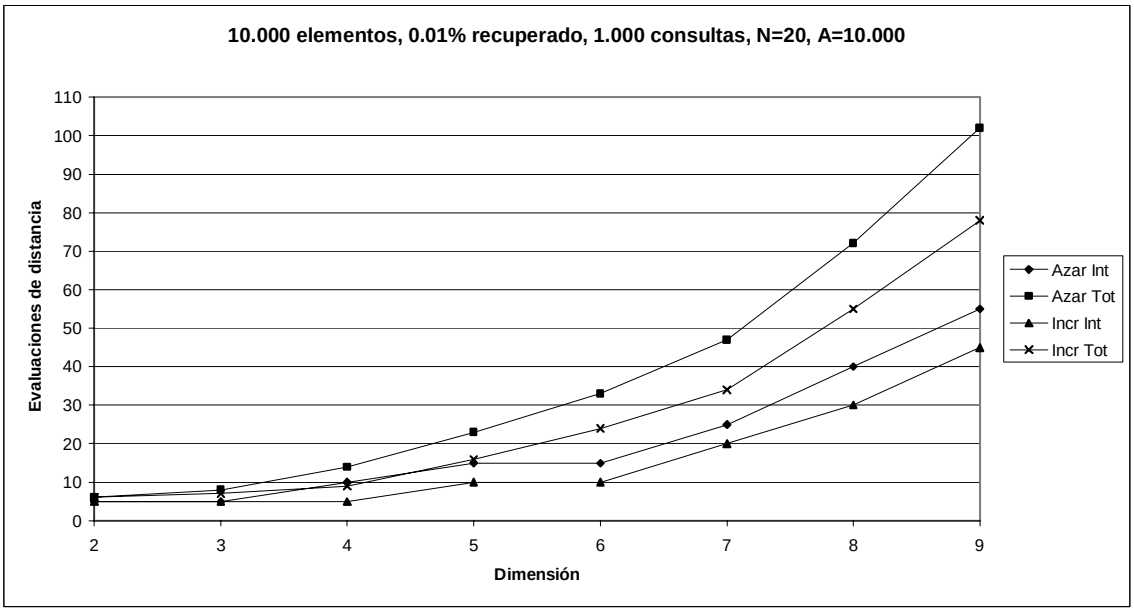

Gráfico 6-10: Número óptimo de pivotes usando métodos de selección aleatorio e incremental en espacios de dimensión baja

En muchas aplicaciones prácticas no se dispone de la suficiente memoria para alcanzar el número óptimo de pivotes. Los gráficos 6-11 y 6-12 muestran esta situación, comparando el rendimiento entre elegir los pivotes aleatoriamente y utilizar el método incremental de selección cuando se dispone de una cantidad fija de pivotes. En este experimento se mantuvo constante el número de elementos del espacio vectorial y se varió la dimensión entre 10 y 20.

Se aprecia de ambos gráficos que a medida que aumenta la dimensión del espacio disminuye la brecha entre ambos métodos de selección de pivotes, y que en espacios de dimensión alta es poco lo que se mejora con respecto al método aleatorio, un 9.26% al utilizar 50 pivotes, mientras que en espacios de dimensión más baja la mejora en el rendimiento es casi de un 50%. Este decrecimiento en la mejora del

rendimiento ocurre principalmente por la maldición de la dimensionalidad: a medida que la dimensión del espacio crece se hace cada vez más difícil responder una consulta por rango.

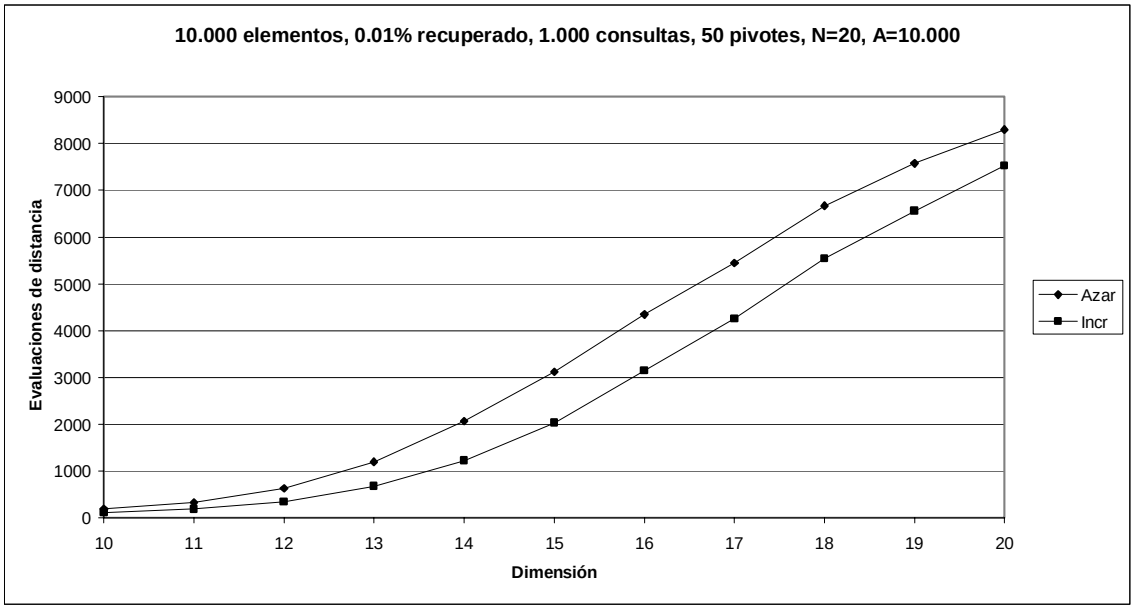

Gráfico 6-11: Complejidad total de los métodos aleatorio y selección incremental de pivotes utilizando $k = 50$ pivotes

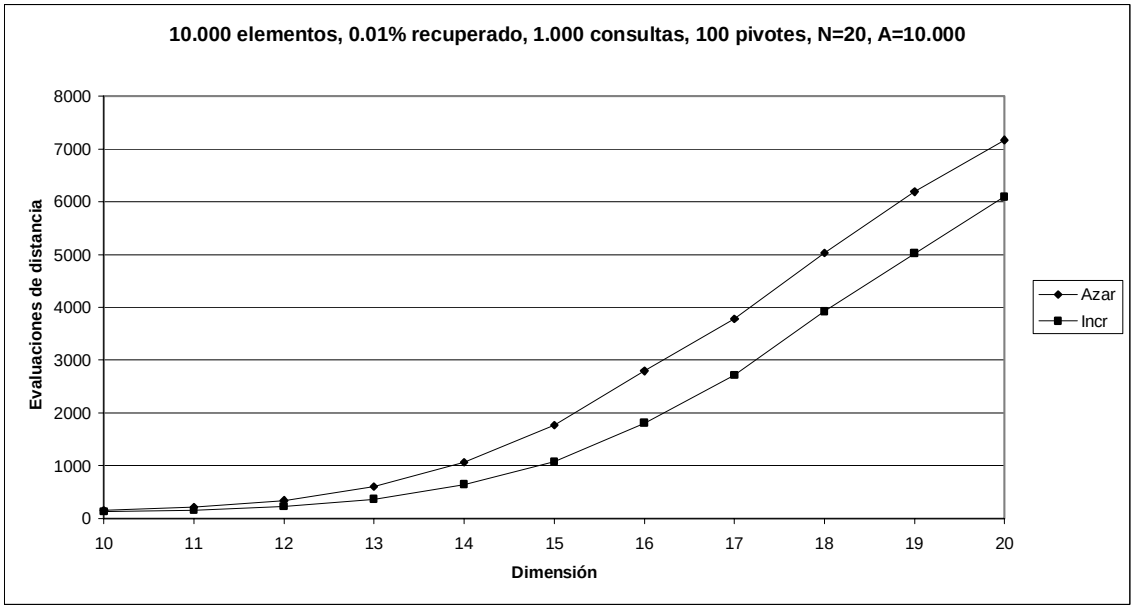

Gráfico 6-12: Complejidad total de los métodos aleatorio y selección incremental de pivotes utilizando $k = 100$ pivotes

El gráfico 6-13 muestra el resultado de comparar costo versus número de pivotes necesarios para obtener dicho costo en un espacio vectorial de dimensión 16. El número óptimo de pivotes al escogerlos aleatoriamente es de $k^* = 500$ pivotes, mientras que con el método incremental dicho óptimo baja a $k^* = 400$ pivotes (gráfico 6-7). Se aprecia del gráfico que el ahorro máximo de memoria se obtiene cuando el costo asociado al método incremental es igual al costo que se obtiene con el método aleatorio en su número óptimo de pivotes.

El gráfico 6-14 muestra el resultado del mismo experimento anterior pero en un espacio vectorial de dimensión 24. El resultado es más o menos idéntico al caso anterior: si se está dispuesto a tener la misma complejidad externa que el método aleatorio utilizando el máximo número de pivotes posible, el ahorro en memoria es apreciable, en este caso con un costo de 6.700 e.d. (aproximadamente), el ahorro en memoria utilizando el método incremental de selección es de un 28.57%.

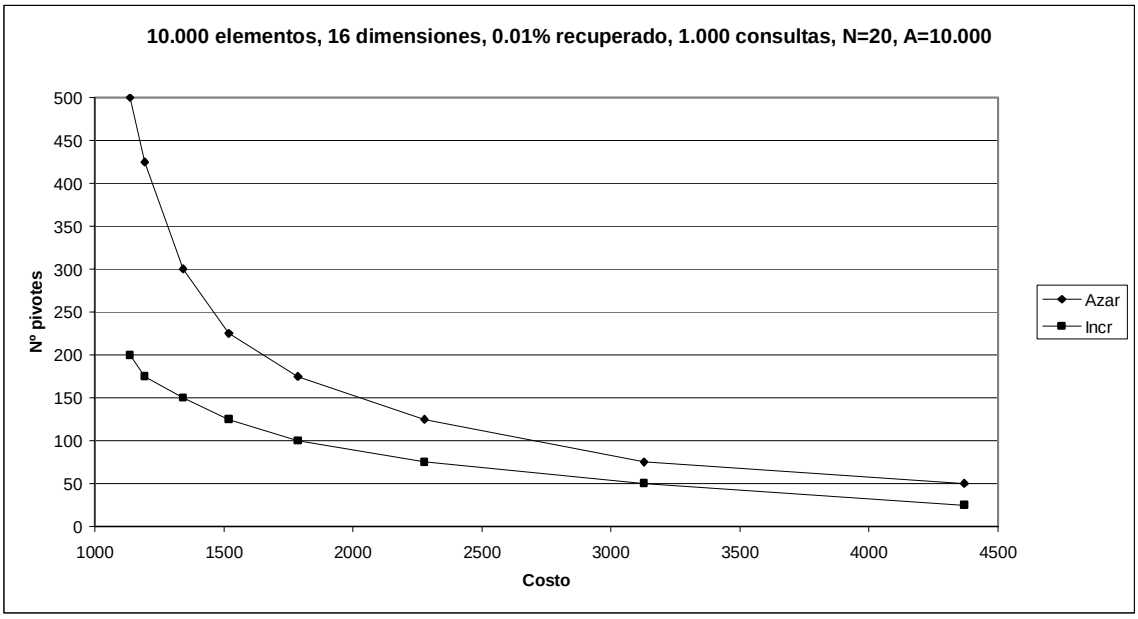

Gráfico 6-13: Nº de pivotes necesarios para responder una consulta por rango con costo C en un espacio de 16 dimensiones

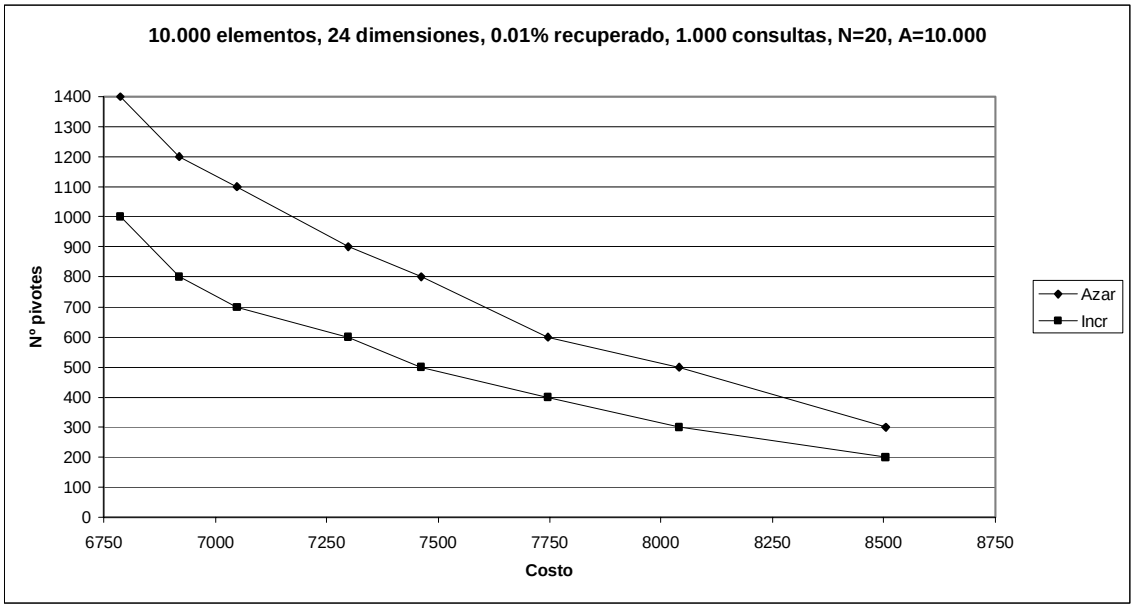

Gráfico 6-14: Nº de pivotes necesarios para responder una consulta por rango con costo C en un espacio de 24 dimensiones

Por último, el gráfico 6-15 muestra la comparación de rendimiento entre los métodos de selección de pivotes utilizando el número óptimo de pivotes por método, al hacer crecer el número de elementos del espacio vectorial, y manteniendo su dimensión constante. Resulta interesante notar que con el método de selección incremental el número óptimo de pivotes prácticamente no aumenta durante un gran rango de número de elementos del espacio, entre 55.000 y 80.000 elementos.

Usando el método de los mínimos cuadrados, se obtienen las siguientes estimaciones para las curvas de complejidad total:

- $y = cn^{\alpha}$:
  - Método de selección aleatorio:
    - $\alpha = 0.51559139 \pm 0.00445407$ (Error porcentual: 0.86%).
    - $c = 0.61520754 \pm 0.03021538$ (Error porcentual: 4.91%).
  - Método de selección incremental:
    - $\alpha = 0.52641427 \pm 0.00688267$ (Error porcentual: 1.31%).
    - $c = 0.41434834 \pm 0.03186441$ (Error porcentual: 7.69%).

- $y = c\log(n) + d$:
  - Método de selección aleatorio:
    - $c = 73.802411 \pm 3.50373261$ (Error porcentual: 4.75%).
    - $d = -628.245738 \pm 37.7161034$ (Error porcentual: 6.00%).
  - Método de selección incremental:
    - $c = 57.0063741 \pm 2.80540442$ (Error porcentual: 4.92%).
    - $d = -487.607985 \pm 30.198915$ (Error porcentual: 6.19%).

La cota inferior mostrada en la ecuación 4-3 predice un comportamiento logarítmico en función del número de elementos del espacio, sin embargo el modelo $y = cn^{\alpha}$ para este caso tiene un menor error porcentual que el modelo $y = c\log(n) + d$.

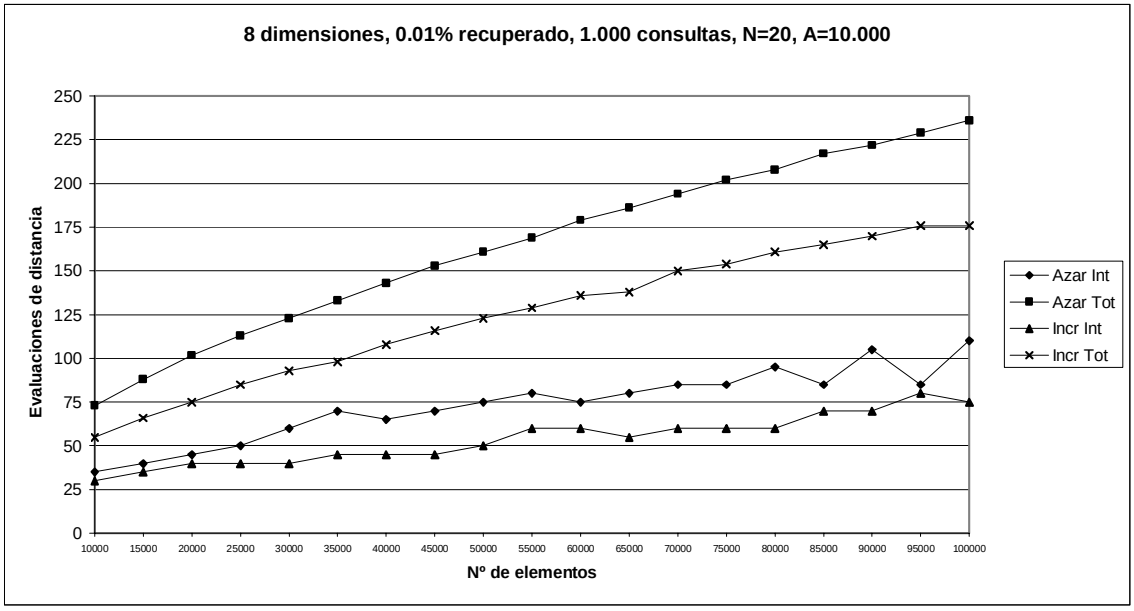

Gráfico 6-15: Complejidad interna y total de los métodos aleatorio y selección incremental de pivotes, utilizando el número óptimo de pivotes según cantidad de elementos del espacio

6.4 Características de un buen conjunto de pivotes

En las tablas 6-1, 6-2 y 6-3 se muestra estadísticas de las distancias entre elementos de un espacio vectorial de 10.000 elementos dimensión 8, 16 y 24, respectivamente, y con respecto a un "buen conjunto de pivotes" escogidos utilizando el método incremental de selección, con $N = 20$ y $A = 10.000$. Se utilizaron las fórmulas estadísticas clásicas en el cálculo de la media y la varianza de los datos medidos.

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.127746	1.402587	1.262141
Varianza σ^2	0.060189	0.057650	0.062895

Tabla 6-1: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección incremental en un espacio de dimensión 8, con 30 pivotes

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.611856	1.827125	1.722515
Varianza σ^2	0.059467	0.063944	0.061440

Tabla 6-2: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección incremental en un espacio de dimensión 16, con 400 pivotes

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.983718	2.051986	2.017805
Varianza σ^2	0.059106	0.061541	0.060622

Tabla 6-3: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección incremental en un espacio de dimensión 24, con 1800 pivotes

Los datos obtenidos muestran dos aspectos interesantes a considerar sobre las características de un buen conjunto de pivotes:

- Los pivotes están **más alejados entre sí** que el promedio de distancias dentro el conjunto de elementos. Esto indica que la lejanía entre pivotes entrega más información al realizar una consulta por rango que si los pivotes estuvieran cercanos, lo cual es lógico de suponer puesto que dos pivotes muy cercanos entre sí entregarían casi la misma información sobre el conjunto de elementos, y por lo tanto uno de ellos no sería un gran aporte al momento de realizar la consulta.
- Los pivotes están **más alejados al resto de los elementos del conjunto** que el promedio de distancias dentro de éste.
- La varianza para las estadísticas mostradas varía ligeramente al considerar los pivotes escogidos.
- Las diferencias de las medias de distancias en los distintos casos disminuye a medida que la dimensión del espacio aumenta.

Los tres primeros puntos anteriores señalan que los buenos pivotes son aquellos elementos que se encuentran alejados entre sí y alejados del resto del conjunto de elementos, es decir, son elementos **aislados** dentro del conjunto. A estos elementos se les denomina *outliers*.

El cuarto punto es una muestra de cómo la maldición de la dimensionalidad hace cada vez más difícil encontrar buenos conjuntos de pivotes, ya que cuando el espacio es de dimensión alta los elementos ya se encuentran alejados entre sí, por lo que es más difícil discriminar y elegir aquellos elementos “aislados” dentro del conjunto.

6.5 Un método alternativo de selección de pivotes

Si una característica de los buenos pivotes es que son *outliers*, se puede suponer que los *outliers* de por sí son buenos pivotes, por lo que un nuevo método de selección de k pivotes sería el siguiente:

- Elegir el primer pivote al azar.
- Se dispone de un conjunto de pivotes $\{p_1, p_2, ..., p_{i-1}\}$, y se desea elegir el i-ésimo pivote. De una muestra de tamaño X del conjunto de elementos, se elige como pivote aquel elemento que maximice la suma de distancias a los pivotes previamente elegidos.
- El proceso termina cuando ya se han escogido k pivotes.

A este método se le denominará **método incremental de selección de *outliers* (Outl)**, y su costo asociado es:

- Elegir el primer pivote tiene costo 0, pues se elige al azar.
- Para elegir el i-ésimo pivote, por cada candidato hay que calcular la distancia a los pivotes previamente elegidos, es decir se necesitan $i-1$ evaluaciones de la distancia d. Como son k elementos candidatos, el costo total en la i-ésima ietración es de $(i-1)X$.
- El proceso se repite k veces, por lo que el trabajo total es:

$$Trabajo = 0 + X + 2X + 3X + ... + (k-1)X = \frac{k(k-1)}{2} X \text{ evaluaciones de distancia.}$$

Dado que los métodos de selección de pivotes vistos en la sección 5.3 tienen un costo de $2NAk$ evaluaciones de distancia, para que el nuevo método realice el mismo trabajo debe cumplirse que $X = \frac{4NA}{k-1}$.

Se realizó el mismo experimento descrito en la sección 6.2, comparando el método de selección incremental de *outliers* con el método incremental de selección de pivotes en espacios de 8, 16 y 24 dimensiones. Los resultados obtenidos se muestran en los gráficos 6-16, 6-17 y 6-18.

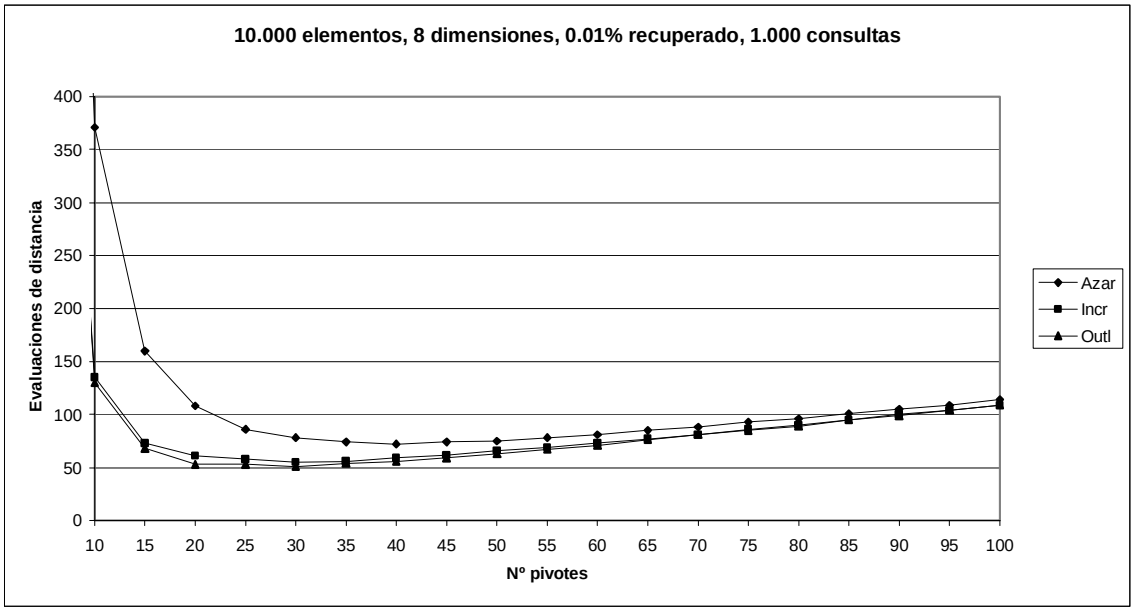

Gráfico 6-16: Comparación entre método incremental y *outliers* en espacio de 8 dimensiones

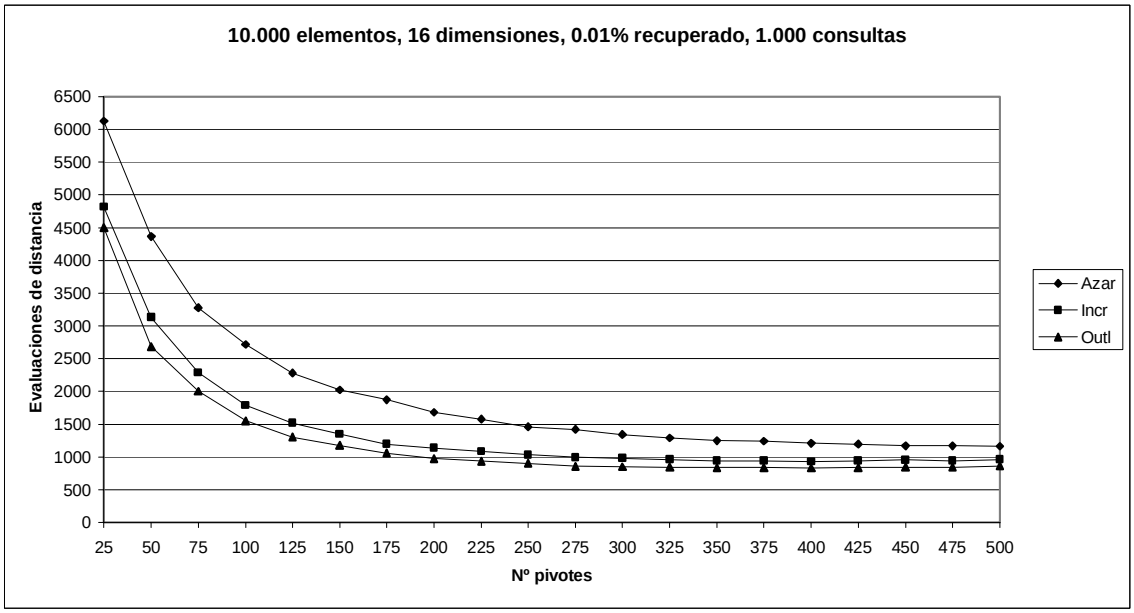

Gráfico 6-17: Comparación entre método incremental y *outliers* en espacio de 16 dimensiones

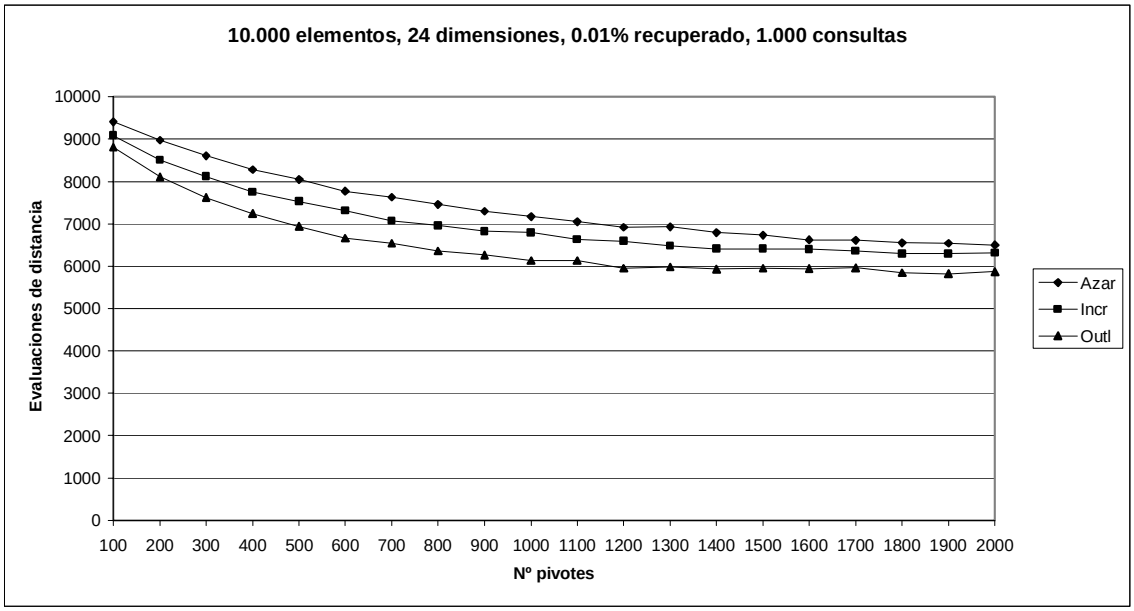

Gráfico 6-18: Comparación entre método incremental y *outliers* en espacio de 24 dimensiones

Se observa de los gráficos anteriores que, a medida que la dimensión del espacio métrico aumenta, el método de selección de *outliers* posee un mejor rendimiento que el método incremental de pivotes en la práctica. Es necesario notar, eso sí, que el método de selección de *outliers* **no maximiza la media del espacio de pivotes** a medida que la dimensión del espacio crece, como lo muestran los resultados mostrados en la tabla 6-4.

	$\mu_{p_{INCR}}$	$\mu_{p_{OUTL}}$
8 dimensiones	0.756291	0.783124
16 dimensiones	1.077653	1.055191
24 dimensiones	1.659382	1.415276

Tabla 6-4: Comparación de la media de la distancia D entre conjuntos de pivotes obtenidos con el método de selección incremental y el método de selección de *outliers*

Esto quiere decir que si bien los pivotes elegidos al maximizar la media de D son *outliers*, elegir un conjunto de *outliers* no necesariamente implica maximizar la media de D.

Las estadísticas de las distancias entre elementos y *outliers* se muestran en las tablas 6-5, 6-6 y 6-7, respectivamente en espacios de 8, 16 y 24 dimensiones. Se observa para este caso que la diferencia de la media de la distancia entre elementos y entre los pivotes es mucho más pronunciada que al elegir los pivotes con el método de selección incremental.

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.127746	1.554554	1.345091
Varianza σ^2	0.060189	0.080980	0.067738

Tabla 6-5: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección de *outlier*s en un espacio de dimensión 8, con 30 pivotes

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.611856	1.979223	1.805184
Varianza σ^2	0.059467	0.066654	0.059718

Tabla 6-6: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección de *outliers* en un espacio de dimensión 16, con 400 pivotes

	Distancia entre elementos del espacio	Distancia entre pivotes	Distancia entre elementos del espacio y pivotes
Media μ	1.983718	2.235076	2.112510
Varianza σ^2	0.059106	0.057398	0.057134

Tabla 6-7: Estadísticas de las distancias entre elementos del conjunto y los pivotes escogidos con el método de selección de *outliers* en un espacio de dimensión 24, con 1900 pivotes

Los resultados obtenidos en los gráficos 6-16, 6-17 y 6-18 pueden llevar a pensar que la heurística de elegir *outliers* como pivotes puede ser buena, y de hecho es una técnica recomendada por algunos autores [15,24], pero que no tiene un fundamento matemático concreto como en el caso de maximizar la media de D. De hecho, maximizar la media de D corresponde a maximizar la varianza de distancias en el espacio original. Esto es lo que hace el método incremental de selección, mientras que el método de selección de *outliers* maximiza la media de distancias en el espacio original. Ambas cosas no siempre van juntas, es decir, el hecho de maximizar una no implica necesariamente maximizar la otra.

7. Utilización del método de selección de pivotes en ejemplos de la vida real

A continuación se presentan tres comparaciones entre elegir los pivotes aleatoriamente, utilizando el método incremental de selección y utilizando el método de selección de *outliers*, solo que ahora los elementos de los espacios métricos no son elegidos uniformemente distribuidos, sino que son elegidos con distribuciones especiales o son datos de ejemplos extraídos de aplicaciones reales.

En el caso de las secciones 7.1 y 7.2 los conjuntos de datos y programas fuentes usados fueron obtenidos desde la siguiente URL: http://www.dimacs.rutgers.edu/Challenges/Sixth/software.html.

Estos ejemplos muestran que el método de selección incremental efectivamente elige buenos conjuntos de pivotes para una gama de aplicaciones reales de los algoritmos de búsqueda en proximidad basados en pivotes, y que el método de selección de *outliers* no funciona bien en todas las aplicaciones de la vida real.

7.1 Imágenes de la NASA

Se escogieron 40.700 imágenes de los archivos de fotografías y videos de la NASA. En el caso de los videos, se escogieron cortes especiales basados en la transición del histograma de colores, obteniéndose imágenes representativas. A través de un proceso de conversión basado en el histograma de colores de cada imagen, se obtuvo una representación vectorial en 36 dimensiones, y usando un análisis de componentes principales se redujo la dimensión del vector a 20 dimensiones, en donde los valores de las coordenadas están en el rango $(-1,1)$. La distancia entre imágenes se calcula como la distancia euclidiana L_2 entre los vectores obtenidos en el proceso de conversión.

Del total de imágenes escogidas, 37.000 corresponden al conjunto de elementos del espacio, y las 3.700 imágenes restantes corresponden a elementos de consulta en dicho espacio. Se utilizó un radio de búsqueda tal que cada consulta por rango devolviera en promedio el 0.1% del total de imágenes del espacio. Los resultados obtenidos se promediaron sobre 1.000 consultas realizadas, elegidas del conjunto de consulta.

En el método de selección incremental se utilizaron los parámetros con valores $N = 20$ y $A = 10.000$. La comparación entre éste método con la selección aleatoria de pivotes y la selección de *outliers* se muestra en el gráfico 7-1.

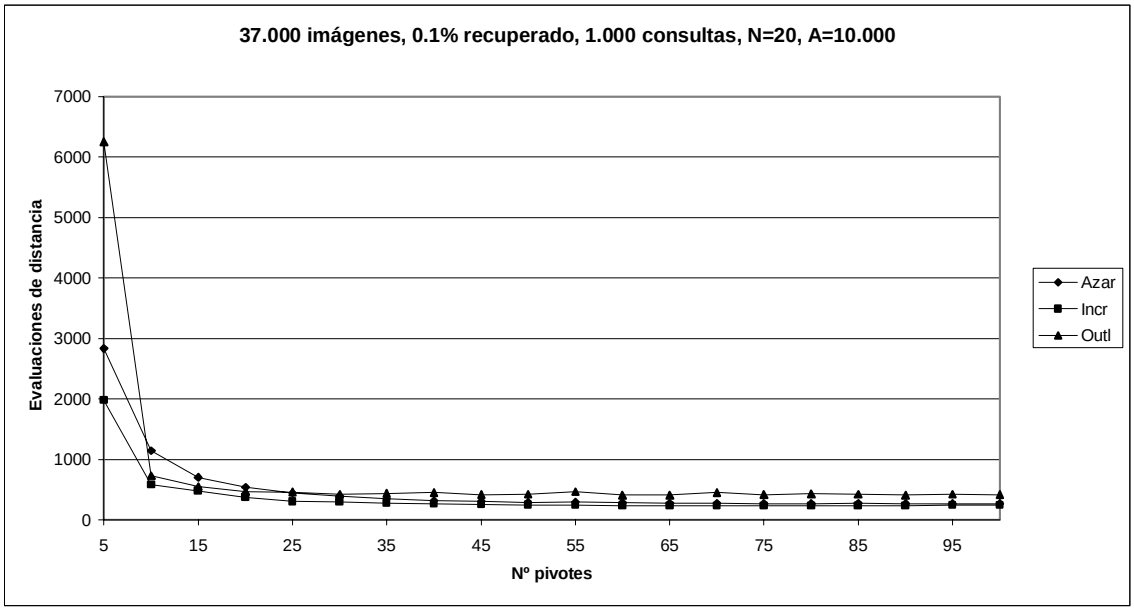

Gráfico 7-1: Complejidad total del algoritmo de búsqueda en el espacio de imágenes

Se observa del gráfico que el número óptimo de pivotes para el método aleatorio e incremental es muy parecido, 75 y 80 pivotes respectivamente, y que el número de e.d. en el óptimo es menor en el caso del método incremental, 234 e.d. versus 269 e.d del método aleatorio, lo que implica una mejora del 13.01%.

Al mismo costo que en el óptimo con la elección aleatoria de pivotes, el método incremental sólo requiere de 40 pivotes, lo que implica un ahorro en memoria del 50%.

El método de selección de *outliers* se comporta peor que el método de selección incremental en el intervalo estudiado, y a veces incluso peor que el método aleatorio.

Un aspecto interesante de este espacio es que si se calcula el radio necesario para recuperar en promedio el 0.01% del total de elementos del conjunto, se obtiene que dicho radio es muy parecido al utilizado para recuperar la misma fracción de elementos en un espacio vectorial uniformemente distribuido de dimensión 5, lo que querría decir que en realidad la dimensión intrínseca del espacio de imágenes es mucho menor

que 20. En efecto, al calcular la dimensión intrínseca del espacio de imágenes, utilizando la distancia L_2, se obtuvo que $\rho_{imágenes} = 5.16$.

7.2 Espacio vectorial con distribución de Gauss

Se creó un espacio vectorial con 10.000 elementos de dimensión 10, pero las coordenadas de los vectores ya no son uniformemente distribuidas en el cubo unitario, sino que son valores reales con una distribución de Gauss cuya media es un valor real aleatorio en el intervalo $[0,10]$ y con varianza igual a 1.0. De la misma manera se generaron 1.000 puntos para ser ocupados como consultas en dicho espacio.

Se promediaron los resultados sobre las 1.000 consultas realizadas, y se utilizó un radio de búsqueda tal que las consultas retornaran en promedio el 0.01% del total del conjunto de elementos. En el método incremental se utilizaron los parámetros $N = 20$ y $A = 10.000$.

Los resultados de la comparación entre elegir conjuntos de pivotes aleatoriamente versus utilizar el método incremental de selección y el método de selección de *outliers* se muestran en el gráfico 7-2.

En este experimento el óptimo con la selección aleatoria es de 55 pivotes, mientras que el óptimo con el método incremental es de 50 pivotes. El costo total en el óptimo con el método aleatorio es de 167 e.d., mientras que en el óptimo del método incremental el costo total es de 165 e.d., es decir, casi igual que con la elección aleatoria de pivotes. A mismo costo en el óptimo del método aleatorio, el método incremental requiere de 45 pivotes, lo que implica un ahorro en memoria del 18.18%. La dimensión intrínseca de este espacio métrico, calculado con la distancia L_2, resultó ser $\rho_{Gauss1} = 4.47$. En este caso el método de selección de *outlier*s funcionó un poco mejor que el método incremental de selección de pivotes.

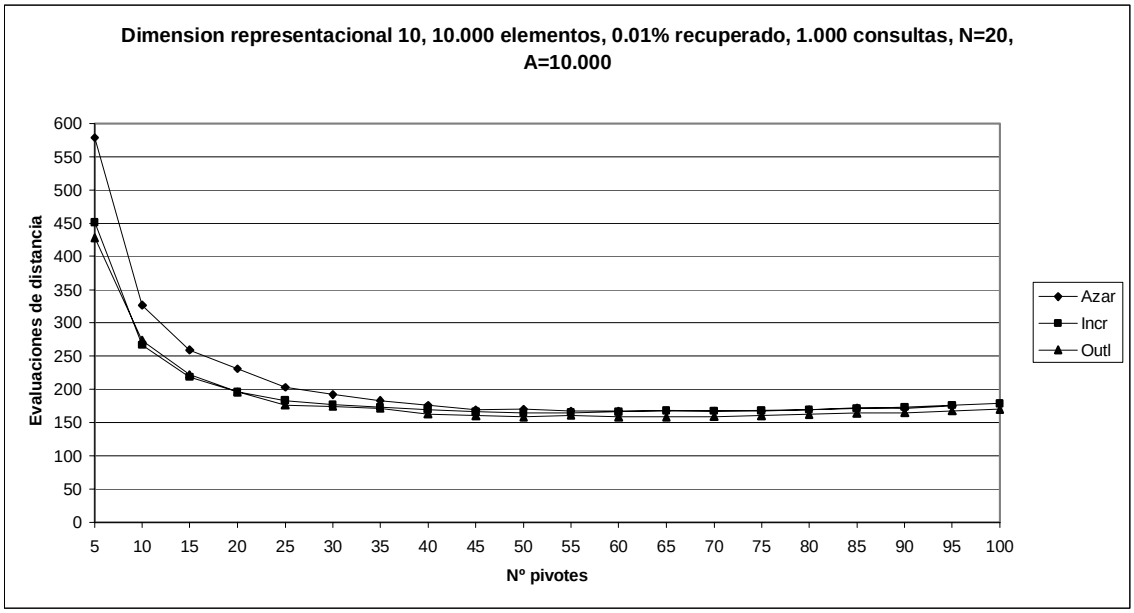

Gráfico 7-2: Complejidad total del algoritmo de búsqueda en el espacio vectorial con varios *clusters*, coordenadas con distribución de Gauss

Se repitió el mismo experimento anterior, pero se aumentó la dimensión del espacio vectorial a 30 y se cambió el valor de la varianza de las coordenadas a 0.1, es decir, la dimensión intrínseca del espacio generado es mucho menor a la dimensión representacional del espacio. En efecto, se obtuvo que la dimensión intrínseca de este espacio es de $\rho_{Gauss\,2} = 4.74$, menos de un sexto de la dimensión representacional.

Los resultados obtenidos para este experimento se muestran en el gráfico 7-3. En este caso se observa que el método incremental y el método outliers tienen un mejor rendimiento que el método aleatorio. El óptimo con el método aleatorio se alcanza con 16 pivotes, mientras que el óptimo con el método incremental se alcanza con 6 pivotes. El costo en ambos óptimos es prácticamente igual (mejora un 1% con el método incremental), por lo que el ahorro en memoria a igual costo es de un 62,5%.

El método incremental de selección de pivotes mejoró el rendimiento del algoritmo en ambos experimentos.

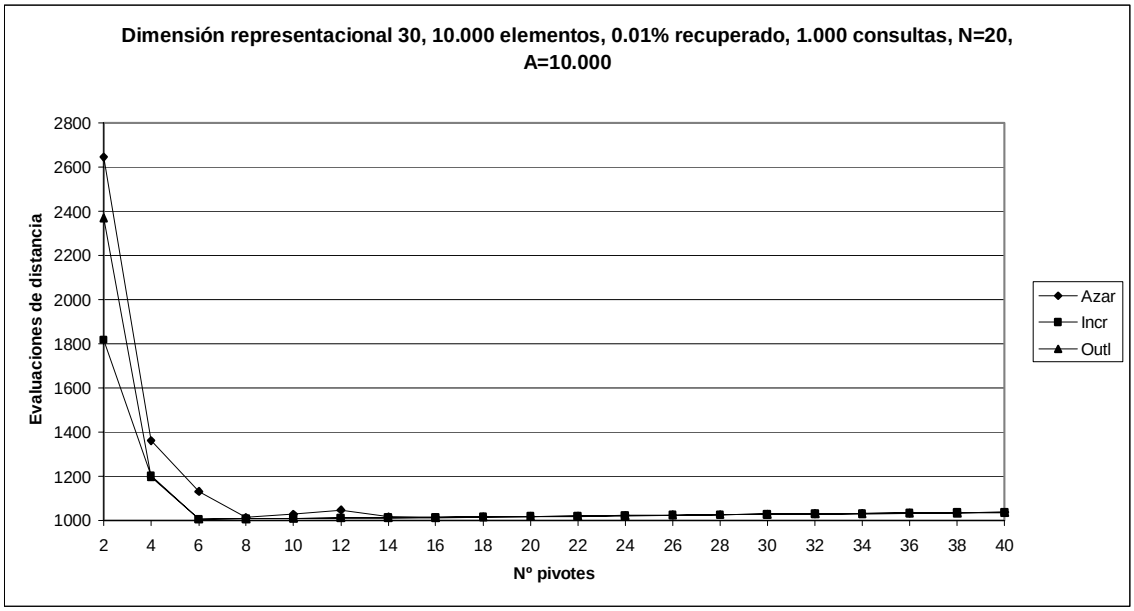

Gráfico 7-3

7.3 Espacio de strings

El espacio métrico de los *strings* (cadenas de caracteres) es muy diferente a todos los espacios métricos vistos hasta el momento, puesto que en este caso la métrica de dicho espacio, la **distancia de edición**, es una función discreta que mide la mínima cantidad de inserciones, borrados y reemplazos de caracteres necesarios para que dos *strings* sean iguales. Además, en el espacio de *strings* no se posee información adicional sobre las coordenadas de los puntos, como en el caso de los espacios vectoriales.

De un diccionario español con 86.601 palabras, se escogieron 77.455 para conformar el espacio métrico, y las restantes 8.606 palabras se utilizaron como conjunto de elementos de consulta. Se eligieron conjuntos de pivotes con el método aleatorio y utilizando el método incremental de selección, y se promediaron los resultados sobre 1.000 consultas realizadas. El gráfico 7-4 muestra los resultados del experimento realizado utilizando un radio $r = 1$, que retorna en promedio un 0.00244% del total del conjunto de palabras.

Se observa del gráfico que el método incremental mejora el rendimiento del algoritmo de búsqueda en proximidad al utilizar pocos pivotes. El número óptimo de pivotes con el método aleatorio es de 45 pivotes, mientras que el número óptimo con el método de selección incremental es de 35 pivotes. El rendimiento en el óptimo para ambos métodos es casi el mismo, el método de selección incremental

mejora el total de e.d. en sólo un 4.84%. El ahorro en memoria al mismo costo que en el óptimo del método aleatorio, utilizando el método incremental de selección, es de 22.22%.

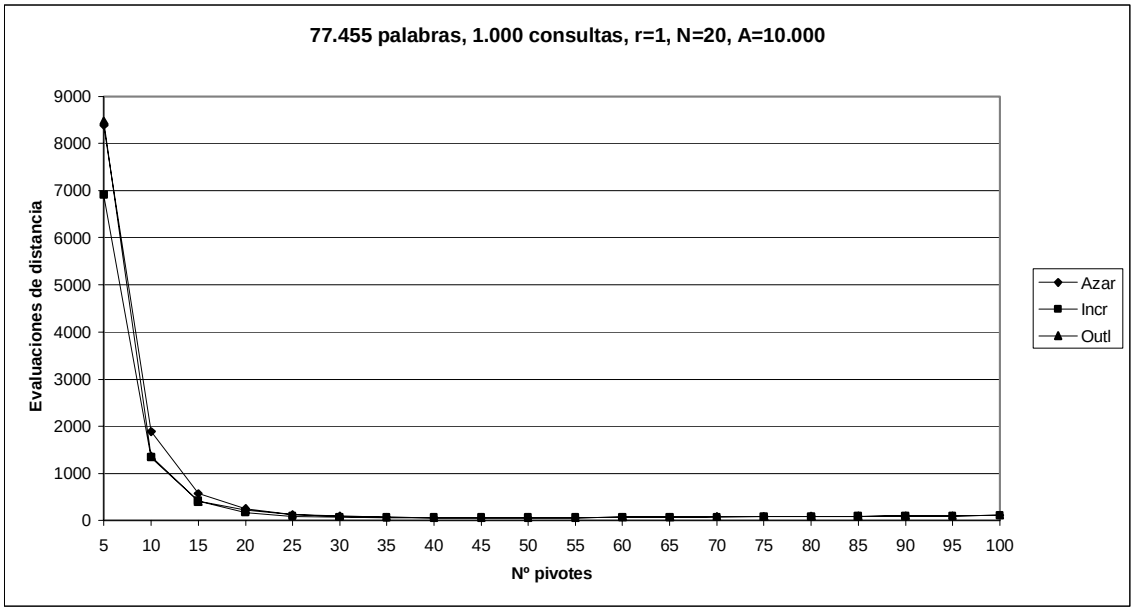

Gráfico 7-4: Complejidad total del algoritmo de búsqueda en el espacio de *strings*, $r = 1$

El gráfico 7-5 muestra el resultado de un experimento idéntico al anterior, pero ahora con un radio de búsqueda $r = 2$, que retorna en promedio un 0.02967% del total de palabras del conjunto.

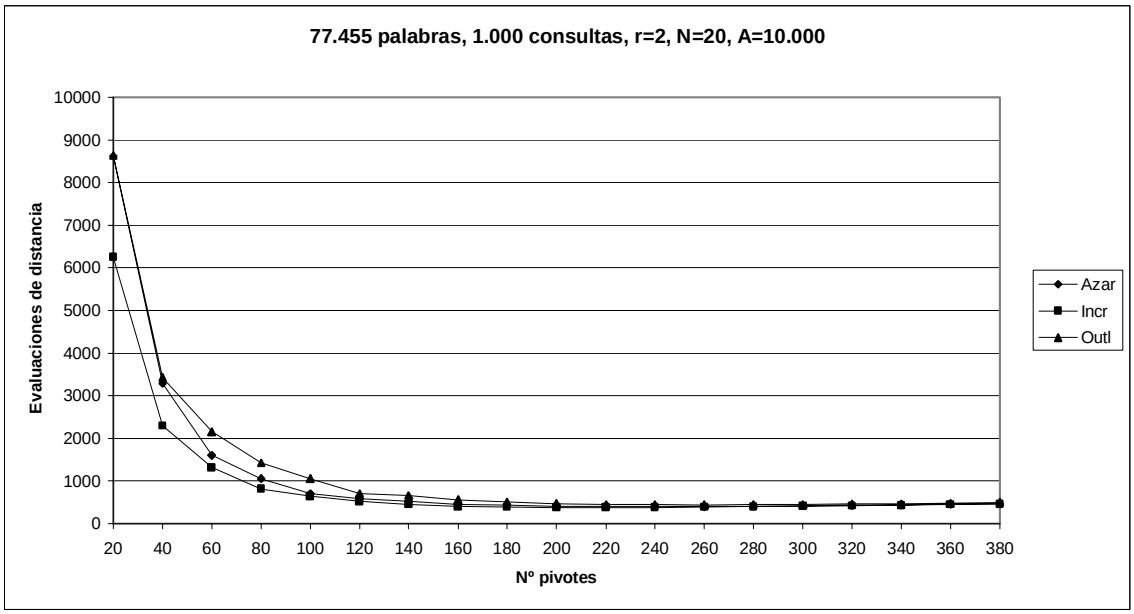

Gráfico 7-5: Complejidad total del algoritmo de búsqueda en el espacio de *strings*, $r = 2$

Para este experimento, el óptimo número de pivotes con el método aleatorio es de 260 pivotes, mientras que con el método incremental de selección el óptimo se alcanza con 200 pivotes. En el óptimo número de pivotes con el método aleatorio se tiene un costo de 401 e.d., mientras que en el óptimo con el método incremental se tiene un costo de 375 e.d., lo que implica una mejora en el rendimiento del algoritmo de búsqueda del 6.48%. Al mismo costo que en el óptimo con el método aleatorio, el método incremental solo requiere de 160 pivotes, lo que implica un ahorro en memoria del 38.46%. Se obtuvo que la dimensión intrínseca del espacio de *strings* es de $\rho_{Strings} = 8.73$.

En ambos experimentos, el algoritmo de búsqueda en proximidad mejora su rendimiento al usar el método incremental de selección de pivotes. El método de selección de *outliers* obtuvo un rendimiento peor que al elegir los pivotes con el método aleatorio.

8. Conclusiones y recomendaciones generales

El problema de cómo elegir buenos conjuntos de pivotes ha sido tema de discusión en el pasado, pero hasta el momento no se habían propuesto métodos formales de selección de pivotes, sólo se habían propuesto heurísticas con argumentos vagos. Este estudio propone una función objetivo a maximizar basada en las características que poseen los espacios métricos, y después se obtiene abundante evidencia empírica que respalda la hipótesis que, tanto en casos sintéticos como en ejemplos de la vida real, maximizar dicha función objetivo es una buena idea para generar buenos conjuntos de pivotes.

Los principales resultados obtenidos en el presente trabajo se resumen en los siguientes puntos:

1. Se definió a partir del histograma de distancias del espacio métrico y del espacio definido por el conjunto de pivotes un criterio de eficiencia para elegir buenos pivotes, el cual consiste en maximizar la media de la distancia entre elementos mapeados al espacio de pivotes.

2. Se mostraron distintos métodos para elegir conjuntos de buenos pivotes, y se mostró a través de resultados experimentales que el método de selección incremental de pivotes obtiene mejores resultados en la práctica realizando el mismo trabajo de indexación. También se analizó la influencia de los valores de los distintos parámetros de los métodos de selección de pivotes, y se concluyó que es preferible aumentar el número de pares de elementos con el cual se estima la media de distancias en el espacio de pivotes antes que aumentar el tamaño de la muestra desde donde se elige un nuevo pivote.

3. Se mostró experimentalmente que la técnica de selección propuesta efectivamente elige buenos conjuntos de pivotes, con los cuales la complejidad total del algoritmo de búsqueda en proximidad disminuye, que el número óptimo de pivotes disminuye con respecto al número óptimo al elegir los pivotes aleatoriamente, y que el tamaño del índice cuando los pivotes son elegidos con el método de selección propuesto es menor al tamaño del índice con los pivotes elegidos con el método aleatorio al fijar el costo total del algoritmo de búsqueda.

4. Se analizaron las características de un buen conjunto de pivotes, encontrándose que los buenos pivotes son elementos aislados entre sí y del resto de los elementos del espacio, es decir, los buenos pivotes son *outliers*. A partir de esto se estudió un método de selección de pivotes propuesto en [15,24], encontrándose que en los experimentos sintéticos obtuvo mejores resultados que el método de selección incremental, pero que en los ejemplos de la vida real obtuvo peores resultados incluso que al elegir el conjunto de pivotes con el método aleatorio.

5. Se mostró que el método de selección incremental de pivotes elige buenos conjuntos de pivotes en espacios métricos reales: conjuntos de imágenes, espacios vectoriales con elementos organizados en *clusters* y espacio de *strings*. Esto concluye que el criterio de eficiencia definido a partir del histograma de distancias del espacio de elementos se sostiene en distintos tipos de espacios métricos, incluso cuando la métrica de dicho espacio es una función discreta.

A partir de estos resultados, se recomienda utilizar el método incremental de selección de pivotes para generar buenos índices de espacios métricos, pero en dimensiones altas se recomienda utilizar pivotes hasta alcanzar el costo que se obtiene en el óptimo al elegir los pivotes con el método aleatorio, puesto que la mejora en el número de evaluaciones de distancia es mucho menor en comparación con el ahorro en memoria que se obtiene al utilizar pocos pivotes al mismo costo.

El método de selección incremental es directamente aplicable a los algoritmos que utilizan un número fijo de pivotes (FHQT, FQA, LAESA, Spaghettis), y es posible que el criterio de eficiencia definido también funcione en otros algoritmos de búsqueda que utilicen pivotes (VPT, MVPT, BKT).

Una de las conclusiones más interesantes obtenidas durante el desarrollo de este trabajo es haber mostrado que los buenos pivotes son *outliers*, pero que elegir *outliers* no implica necesariamente elegir buenos pivotes. El criterio de eficiencia definido a partir de los histogramas de distancias se sostuvo en todos los espacios métricos en donde se probó el método incremental de selección, y siempre se mejoró la complejidad total del algoritmo de búsqueda en proximidad.

Esto conlleva a discutir la manera en como se plantean las heurísticas de selección de pivotes: no necesariamente una heurística que funcione bien en un espacio métrico en particular va a funcionar bien en todos los espacios métricos. En particular, elegir *outliers* como buenos pivotes es una heurística que ya había sido recomendada por algunos autores [15,24] y justificada con algunos ejemplos particulares. Sin embargo, los resultados experimentales muestran que no siempre esta heurística mejora el rendimiento del algoritmo de búsqueda, y que incluso a veces funciona peor que al elegir los pivotes con el método aleatorio.

Lo anterior también plantea una nueva interrogante: ¿Es válido probar métodos de selección de pivotes en espacios vectoriales sintéticos, con elementos uniformemente distribuidos y consultas al azar? La ventaja de realizar los experimentos con espacios vectoriales tratados como espacios métricos (sin utilizar la información de las coordenadas, solamente la función de distancia) es que se puede fijar con precisión la

dimensión del espacio métrico con el cual se desea trabajar, pero, como se vio con el método de selección de *outliers*, también puede llevar a conclusiones empíricas erróneas.

Otro tema que queda abierto a discusión es cómo adaptar el método incremental de selección de pivotes para utilizarlo en algoritmos de búsqueda basados en *clustering*. Se puede conjeturar que el método no debiera elegir buenos centros para las *clusters*, puesto que podría elegir un elemento completamente aislado de los *clusters* que existan dentro del espacio métrico, y en ese caso dicho elemento sería un pésimo centro.

9. Referencias

[1] P. Apers, H. Blanken y M. Houtsma. *Multimedia databases in perspective*. Springer, 1997.

[2] D. Sankoff y J. Kruskal, editores. *Time warps, string edits and macromolecules, the theory and practice of sequence comparison.* Addison-Wesley, 1983.

[3] M. Waterman. *Introduction to computational biology*. Chapman and Hall, 1995.

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

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

[6] E. Chávez, J. Marroquín y G. Navarro. *Overcoming the curse of dimensionality.* En European Workshop on Content-based Multimedia Indexing (CBMI'99), páginas 57-64, 1999.

[7] T. Bozkaya, y M. Ozsoyoglu. *Distance-based indexing for high-dimensional metric spaces.* En Proc. ACM SIGMOD International Conference on Management of Data, páginas 357-368, 1997. Sigmod Record 26(2).

[8] P. Yianilos. *Excluded middle vantage point forests for nearest neighbor search*. En DIMACS implementation challenge, ALENEX'99, Baltimore, MD, 1999.

[9] E. Vidal. *An algorithm for finding nearest neighbors in (approximately) constant average time.* Pattern Recognittion Letters, 4:145-157, 1986.

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

[11] E. Chávez, G. Navarro, R. Baeza-Yates y J. Marroquín. *Searching in metric spaces.* Technical report TR/DCC-99-3, Departamento de Ciencias de la Computación, Universidad de Chile. ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/survmetric.ps.gz, 1999.

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

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

[14] R. Myers, S. Myers y R. Walpole. *Probabilidad y estadística para ingenieros, 6ª ed*. Prentice-Hall Hispanoamericana, S.A, páginas 108-110, 1999.

[15] P. Yianilos. *Data structures and algorithms for nearest neighbor search in general metric spaces.* En Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93), páginas 311-321, 1993.

[16] E. Chávez, J. Marroquín y R. Baeza-Yates. *Spaghettis: an array based algorithm for similarity queries in metric spaces*. En Proc. String Processing and Information Retrieval (SPIRE'99), páginas 38-46, IEEE CS Press, 1999.

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

[18] I. Kalantari y G. McDonald. *A data structure and an algorithm for the nearest point problem.* IEEE Transaction on Software Engineering, 9(5), 1983.

[19] F. Dehne y H. Nolteimer. *Voronoi trees and clustering problems.* Information Systems, 12 (2):171-175, 1987.

[20] P. Ciaccia, M. Patella y P. Zezula. *M-tree: an efficient access method for similarity search in metric spaces.* En Proc. Of the 23rd Conference on Very Large Databases (VLDB'97), pág. 426-435, 1997.

[21] G. Navarro. *Searching in metric spaces by spatial approximation*. En Proc. String Processing and Information Retrieval (SPIRE'99), pág. 141-148. IEEE CS Press, 1999.

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

[23] E. Chávez, G. Navarro. *Measuring the dimensionality of general metric spaces.* Technical report TR/DCC-2000-1, Departamento de Ciencias de la Computación, Universidad de Chile, 2000.

[24] S. Brin. *Near neighbor search in large metric spaces.* En Proc. 21st Conference on Very Large Databases (VLDB'95), pág. 574-584, 1995.

Apéndice A: Pseudocódigo de los métodos de selección de pivotes

A.1 Selección aleatoria de pivotes

```
/* E->n == numero de elementos del conjunto E
   E->puntos[i] == i-esimo elemento del conjunto E
   P->puntos[i] == i-esimo elemento del conjunto de pivotes P

   Se eligen k elementos de E al azar, sin repeticiones, y dichos
   elementos conformarán el conjunto de pivotes
*/

Pivotes creaPivotesAzar(Espacio E, int k)
begin
  Pivotes P;
  for i=1 to k
  begin
    azar=random(n);
    while (azar ya haya sido elegido antes)
      azar=random(n);
    P->puntos[i]=E->puntos[azar];
  end
  return P;
end
```

A.2 Selección de N conjuntos aleatorios

```
/* Datos conocidos: N = numero de iteraciones
                    A = pares de elementos para calculo de la media de
D

   Se eligen N conjuntos de pivotes al azar, se elige aquel que
maximice la media de la distancia D, la cual se estima con A pares de
elementos de E
*/

Pivotes creaPivotesSelec(Espacio E, int k)
begin
  Pivotes P, Paux;
  float dim, max;
  P=creaPivotesAzar(E, k);
  max=mediaD(P, A);
  for i=2 to N
  begin
    Paux=creaPivotes(E, k);
    dim=mediaD(P, A);
    if(dim>max)
    begin
      P=Paux
      max=dim;
    end
  end
  return P;
end
```

A.3 Selección incremental

```
/* Datos conocidos: N = numero de iteraciones
                    A = pares de elementos para calculo de la
                        media de D

   Se eligen los pivotes incrementalmente, agregando aquel que
   maximice la media de la distancia D al considerar fijos los
   pivotes previamente elegidos
*/

Pivotes creaPivotesIncr(Espacio E, int k)
begin
  Pivotes P;
  int ungido, selec;
  float dim, max;
  for i=1 to k
  begin
    /* Se elige un candidato inicial */
    selec=random(n);
    while(selec ya haya sido elegido antes)
      selec=random(n);
    P->puntos[i]=E->puntos[selec];
    max=mediaD(P, A);
    ungido=selec;
    for j=2 to N
    begin
      /* Se busca nuevo cantidato */
      selec=random(n);
      while(selec ya haya sido escogido antes)
        selec=random(n);
      P->puntos[i]=E->puntos[selec];
      dim=mediaD(P, A);
      if(dim>max)
      begin
```

```
        max=dim;
        ungido=selec;
      end
    end
    P->puntos[i]=E->puntos[ungido];
  end
  /* Se retorna conjunto de pivotes escogidos */
  return P;
end
```

A.4 Selección de óptimo local

```
/* Datos conocidos: N = numero de iteraciones
                    A = pares de elementos para calculo de la
                        media de D

   Se eligen un conjunto de pivotes al azar. Luego se elige una
   victima, el pivote que menos contribuye al valor de la media de
   D, y se intenta cambiar por otro pivote que tenga una
   contribucion mayor
*/

/* Esta funcion calcula la matriz M con la diferencia de distancia
   entre los A pares de elementos y los k pivotes elegidos
*/

Matriz calculaM(P, A)
begin
  for i=1 to A
  begin
    p1=primer elemento del i-esimo par de elementos;
    p2=segundo elemento del i-esimo par de elementos;
    for j=1 to k
    begin
      pj=P->puntos[j];
      M[i][j]=abs(d(p2, pj)-d(p1, pj));
    end
  end
  return M;
end
```

```
/* Este procedimiento calcula el indice, por cada fila, de la
   columna donde se encuentra el pivote con mayor contribucion y
   el pivote con segunda mayor contribucion
*/

void calculaIndices(Matriz M, int indice[A], int indice2[A])
begin
  float max, max2, aux;
  for i=1 to A
  begin
    max=0.0;
    indice[i]=0;
    max2=0.0;
    indice2[i]=0;
    for j=0 to k
    begin
      aux=M[i][j];
      if (max<aux)
      begin
        max2=max;
        indice2[i]=indice[i];
        max=aux;
        indice[i]=j; /* Marca el indice del valor maximo */
      end
      else if (max2<aux)
      begin
        max2=aux;
        indice2[i]=j; /* Marca el indice del 2º valor maximo */
      end
    end
  end
end
```

```
/* El metodo de selección de pivotes propiamente tal */

Pivotes creaPivotesOptl(Espacio E, int k)
begin
  int indice[A], indice2[A], victima, selec, imaxcontr, ivictima;
  Pivotes P;
  Matriz M;
  float max, max2, aux, contribucion[k], min, contrnp, maxcontr;
  Punto p1, p2, np;
  /* Se elige un grupo de k pivotes al azar */
  P=creaPivotesAzar(E, k);
  /* Se calcula matriz M */
  M=calculaM(P, A);
  /* Se calculan los indices que hay que calcular */
  calculaIndices(M, indice, indice2);
  /* Se itera N veces */
  for veces=1 to N
  begin
    /* Se evalua la contribucion de cada pivote */
    for(i=0; i<A; i++)
    begin
      contribucion[indice[i]]=
         contribucion[indice[i]]+M[i][indice[i]]-M[i][indice2[i]];
    end
    /* Se elige el peor pivote, la victima */
    victima=0;
    min=contribucion[0];
    for j=1 to k
    begin
      if(contribucion[j]<min)
      begin
        victima=j;
        min=contribucion[j];
      end
    end
    maxcontr=0.0;
```

```
imaxcontr=0;
for j=1 to X
begin
  /* Se elige un nuevo pivote reemplazante de la victima */
  selec=random(n);
  while(selec ya haya sido escogido anteriormente)
    selec=random(n);
  /* Se calcula contribucion del nuevo pivote */
  np=E->puntos[selec];
  contrnp=0.0;
  for i=1 to A
  begin
    p1=primero punto del i-esimo par de elementos;
    p2=segundo punto del i-esimo par de elementos;
    aux=abs(d(p2, np)-d(p1, np));
    if (aux>M[i][indice[i]])
    begin
      contrnp=contrnp+aux-M[i][indice[i]];
    end
  end
  if (maxcontr<contrnp)
  begin
    maxcontr=contrnp;
    imaxcontr=selec;
  end
end
/* Si contribucion de nuevo pivote es mayor a la de la
   victima, se agrega nuevo pivote y se quita a la victima */
if (maxcontr>contribucion[victima])
begin
  np=espacio->puntos[imaxcontr];
  pivotes->puntos[victima]=np;
  /* Se actualiza columna victima de matriz M */
  M=actualizarM(P, A);
  /* Se calculan nuevos arreglos de indices */
  calculaIndices(M, indice, indice2);
```

```
    end
  end
  return P;
end
```