Voronoi-Tree: Optimización y Dinamismo

Roberto Uribe Paredes[*]
Depto. de Ingeniería en Computación
Centro de Investigación de la Web
Universidad de Magallanes, Bulnes 01855, Punta Arenas, Chile
(ruribe@ona.fi.umag.cl)

Fabián Granero
Depto. de Ingeniería en Computación
Universidad de Magallanes, Bulnes 01855, Punta Arenas, Chile
(fgranero@ona.fi.umag.cl)

Gonzalo Navarro [**]
Depto. de Ciencias de la Computación
Centro de Investigación de la Web
Universidad de Chile, Blanco Encalada 2120, Santiago, Chile
(gnavarro@dcc.uchile.cl)

Resumen

El *Voronoi-Tree* es una estructura de datos para búsquedas por similaridad en espacios métricos [DN87]. Esta estructura ha demostrado tener buen desempeño en espacios de alta dimensión, sin embargo, es estática, es decir, no esta diseñada para la inserción y eliminación de objetos una vez construida, lo que implica que no puede ser usada en una serie de aplicaciones interesantes.

El presente trabajo describe la propuesta de una versión dinámica del *Voronoi-Tree* con la implementación de las optimizaciones y del método de *Planos Fantasmas* propuestos para el *egnat* en [UN03, Uri05] basado en el *gnat* [Bri95], demostrando que dichos métodos pueden ser implementados en otras estructuras de tipo árbol. Finalmente se demuestra que es posible dar pleno dinamismo a la estructura y ofrecer un método adecuado y de bajo costo para la eliminación, además, manteniendo un buen desempeño en la búsqueda.

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

1. Introducción

1.1. Antecedentes

Uno de los problemas de gran interés en ciencias de la computación es el de "búsqueda por similaridad", es decir, encontrar los elementos de un conjunto más similares a una muestra. Esta búsqueda es necesaria en múltiples aplicaciones, como ser en reconocimiento de voz e imagen, compresión de video, genética, minería de datos, recuperación de información, etc. En casi todas las aplicaciones la evaluación de la similaridad entre

[*]Parcialmente financiado por el Proyecto MECESUP MAG9901, Mineduc, Chile.
[**]Parcialmente financiado por el Nucleo Milenio Centro de investigación de la Web, Proyecto P01-029-F, Mideplan, Chile.

dos elementos es cara, por lo que usualmente se trata como medida del costo de la búsqueda la cantidad de similaridades que se evalúan.

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

El aumento de tamaño de las bases de datos y la aparición de nuevos tipos de datos sobre los cuales no interesa realizar búsquedas exactas, crean la necesidad de plantear nuevas estructuras para búsqueda por similaridad o búsqueda aproximada. Asimismo, se necesita que dichas las estructuras sean dinámicas, es decir, que permitan agregar o eliminar elementos sin necesidad de crearla nuevamente.

1.2. Marco teórico

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

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

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

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

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

El objetivo de los algoritmos de búsqueda es minimizar la cantidad de evaluaciones de distancia realizadas para resolver la consulta. Los métodos para buscar en espacios métricos se basan principalmente en dividir el espacio empleando la distancia a uno o más objetos seleccionados. El no trabajar con las características particulares de cada aplicación tiene la ventaja de ser más general, pues los algoritmos funcionan con cualquier tipo de objeto[CNBYM01].

Existen distintas estructuras para buscar en espacios métricos, las cuales pueden ocupar funciones discretas o continuas de distancia. Algunos son BKTree [BK73], MetricTree [Uhl91], GNAT [Bri95], VpTree [Yia93], FQTree [BYCMW94], MTree [CPZ97], SAT [Nav02].

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

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

Existen dos criterios para delimitar las áreas en las estructuras basadas en clustering, *hiperplanos* y *radio cobertor* (*covering radius*). El primero divide el espacio en particiones de *Voronoi* y determina el hiperplano

al cual pertenece la consulta según a qué centro corresponde. El criterio de radio cobertor divide el espacio en esferas que pueden intersectarse y una consulta puede pertenecer a más de una esfera.

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

El **Voronoi-Tree** o **VT** es una estructura basada principalmente en el diagrama de Voronoi y usa el criterio de radio cobertor para descartar elementos durante la búsqueda. Está basado en la estructura del árbol binario llamado *Bisertor-Tree* o *BST* [KM83], el cual es construido seleccionando dos puntos clave y dividiendo el resto de los puntos de acuerdo a cuál de ellos está más cerca. Este proceso se realiza recursivamente en ambos hijos.

En el *VT* se seleccionan *3* puntos clave para particionar el espacio, cada punto restante es asignado a la clave más cercana, definiéndose así el subárbol de influencia. Cada subárbol es particionado recursivamente. La diferencia con el *Bisertor* es al crear un nuevo nodo, éste debe incluir al padre como su primer objeto. Esto resulta de interés debido a que como existe replica de objetos dentro del árbol, no es la mejor alternativa para la implementación del método de *Planos Fantasmas*.

Para el presente trabajo se implementó una versión genérica del *VT*, donde se eligen k centros para dividir el espacio.

Para la búsqueda por rango en el VT, se compara la consulta con cada centro, se determina que áreas están dentro del rango de influencia y se procede recursivamente en cada uno de esos subárboles, el resto de los centros se descarta.

El dinamismo es poco común en las estructuras para espacios métricos [CNBYM01], sin embargo, algunas estructuras permiten inserciones eficientes una vez construidas. Respecto de la eliminación, resulta particularmente complicada, debido a que las estructuras podrían verse seriamente afectadas y habría que reconstruirlas parcial o totalmente, con el costo que conlleva. Resultados experimentales de estructuras dinámicas pueden encontrarse en [NR02, Uri05].

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

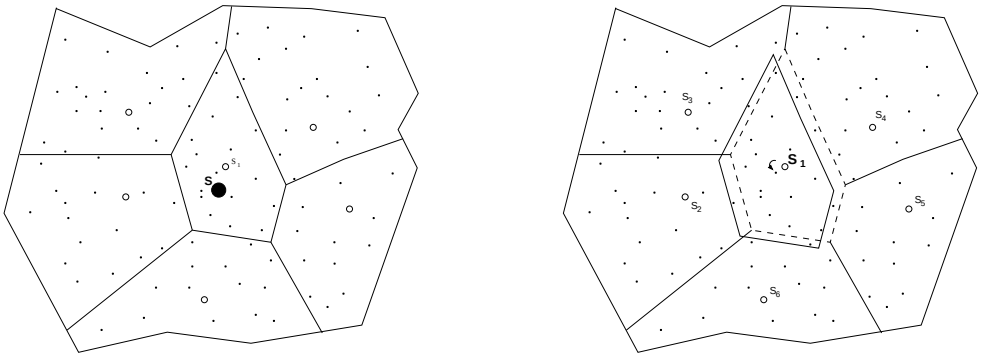
2. Voronoi-Tree

2.1. Construcción del VT

La estructura VT tiene la propiedad que el algoritmo de inserción original al igual que el del *gnat* [Bri95], es en sí dinámico, es decir, como no es necesario conocer a priori la forma del árbol, al insertar un nuevo objeto, este encuentra su lugar, independiente de si la estructura está o no construida anteriormente.

La construcción básica del gnat es como sigue:

1. Se seleccionan k puntos (*centros*), $p_1, \ldots, p_k$ de la base de datos la cual se va a indexar.
2. Se asocia cada punto restante del conjunto de datos al centro más cercano a él. El conjunto de puntos asociados al split p_i se denota como D_{p_i}.
3. Para cada p_i, se calcula el radio cobertor $rc\,(p_i) = max\_d\,(p_i, D_{p_i})$, la máxima distancia $Dist\,(p_i, x)$ donde $x \in D_{p_i}$.



(a) Vista de planos sin eliminaciones

(b) Eliminación de un centro, vista de un *Plano Fantasma*

Figura 1: Voronoi-Tree: partición del espacio, representación de subplanos.

4. Si tuviere que crearse una nueva hoja H para almacenar un nuevo punto p, se debe tomar en cuenta, que al momento de introducir p, también se debe almacenar al centro padre de p en H. Con ésto se puede decir que:

   a) Los árboles de *Voronoi* no tienen hijos excéntricos, esto quiere decir que un hijo no puede tener un radio más grande que su padre.

   b) Un VT es transitivo en el sentido siguiente: $d\left(p, padre(p)\right) = min\left\{(p, p')\,/p' \in antecesores(p)\right\}$

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

Cada conjunto D_{p_i} va a representar un subárbol cuya raíz es p_i, o lo que es lo mismo, cada D_{p_i} va a corresponder al plano de Voronoi cuyo centro es p_i. En la figura 1 (a) se muestra algún momento en la construcción de un plano *P(S)*, cuyo split o centro es S, las divisiones internas corresponderían a los subplanos interiores de S.

2.2. Búsqueda en el Voronoi-Tree

Una búsqueda en un VT, se realiza recursivamente como sigue:

1. Se asume que se desea buscar todos los puntos con distancia $d \leq r$ a el punto x. Sea P la representación del conjunto de puntos del nodo actual (inicialmente la raíz del VT) el cual posiblemente contiene un vecino cercano a x. Inicialmente P contiene todos los puntos split del nodo actual.
2. Se toma un punto p en P, se calcula la distancia *d(x,p)*.
3. Si $d(x,p) - rc(p) > r$, entonces se elimina p de P.
4. Finalmente, se comprueba si: $d(x,p) \leq r$, entonces p pertenece al rango de consulta.
5. Se repiten los pasos 1, 2, 3 y 4 hasta procesar todos los puntos restantes en P.
6. Para todos los puntos $p_i \in P$, se procede recursivamente sobre D_{p_i}.

3. Planos Fantasmas sobre el Voronoi-Tree

3.1. Consideraciones en la eliminación

El objetivo básico es poder tener una estructura que ofrezca total dinamismo y a su vez mantener la eficiencia en las consultas, y ahora reinserciones y eliminaciones. Entonces, en la definición del proceso de eliminación se deben tomar en cuenta ciertas premisas importantes, entre éstas:

1. Después de la eliminación, la estructura debe mantener las mismas características de antes, es decir, ser un *VT*, o contener *VTs*, lo que permitirá reinserciones y búsquedas por rango.

2. No degradar en demasía la eficiencia en las búsquedas, o permitir sólo un determinado aumento en el costo de búsqueda.

Se ha desechado desde el inicio la opción de marcar el elemento o nodo como borrado sin su eliminación física. Esto no es aceptable, debido a que, especialmente en espacios métricos, en la mayoría de las aplicaciones los objetos son muy grandes (por ejemplo imágenes), y es indispensable eliminarlo físicamente. Sin embargo, es posible mantener el nodo sin el objeto, un ejemplo de esto fue propuesto en [NR02] manteniendo *Nodos Ficticios*, para una versión dinámica de la estructura *SAT*. Sin embargo, este método no permite realizar comparaciones donde no existen objetos. El problema de este método es que al haber nodos donde falten algunos objetos, no se puede realizar comparaciones, y por lo tanto habría que incluir dichos subárboles durante todas las búsquedas. Sin embargo, el problema mayor se produce durante la inserción, debido a que se presentan más de una alternativa donde insertar el objeto, lo que implicaría quebrar la forma de construcción del VT.

Descartando inicialmente el caso en que el dato se encuentra en una hoja, lo cual no ofrece complicación en la eliminación, se analizará el caso general, es decir, cuando el objeto se encuentra en un nodo interno del árbol.

Existe una complejidad notoria en la eliminación de un dato en estructuras de tipo árbol. En el Voronoi-Tree existen básicamente tres problemas durante la eliminación:

1. Podrían verse afectados los radios cobertores de los ancestros al dato eliminado, por lo que dichos radios podrían quedar sobre dimensionados.

2. Determinar que hacer con el árbol cuya raíz es el objeto eliminado y el radio cobertor almacenado para éste.

3. Como la estructura replica cada raíz de un subárbol hacia las hojas, implica necesariamente replicar el proceso de eliminación en el caso de eliminar dicha raíz.

3.2. Planos Fantasmas

Una primera alternativa que buscaría mantener la estructura en su forma original es la reconstrucción del subárbol afectado, aquel cuya raíz fue eliminada.

Esto se logra usando un objeto cualquiera como reemplazo y recalculando todas las distancias a partir del nodo donde se encontraba el objeto eliminado. Sin embargo, esto tiene el mismo costo que la reconstrucción del subárbol por reinserción de todos sus descendientes.

La reconstrucción del subárbol puede ser a partir de la raíz o desde el nodo afectado, teniendo la primera la desventaja de aumentar las evaluaciones de distancia, pero con la garantía de que no existirían centros con radio cobertores sobre dimensionados.

La necesidad de reinsertar todos los elementos del subárbol, es para mantener la coherencia de los rangos, sin embargo, es posible encontrar algunos subárboles que podrían insertarse completos, pero no implicaría que se eximieran de algún cálculo de distancia.

Finalmente este método resulta extremadamente costoso debido a la cantidad de evaluaciones de distancia por cada elemento a eliminar.

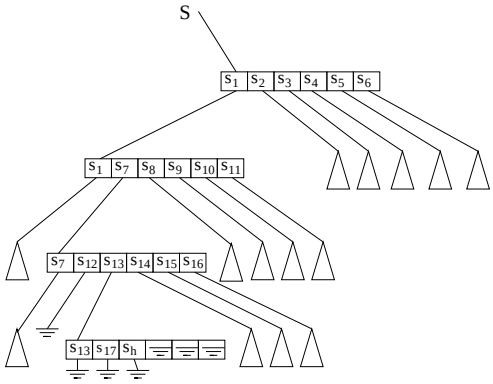


Figura 2: Voronoi-Tree: Estructura original, sin elementos eliminados.

El método de *Planos Fantasmas* propone reemplazar el objeto eliminado por otro que ocupe su lugar en el nodo, es decir, el nuevo objeto sería el centro del plano y conservaría el mismo radio cobertor del elemento eliminado. Como no hay recálculo de rangos, la estructura cambiaría de forma a partir de este nodo produciéndose solapamiento (*overlap*) de los planos, lo cual implica un nuevo elemento a considerar en los métodos de inserción, eliminación y búsqueda.

Este método se denomina *planos fantasmas* debido a que bajo un mismo subárbol coexistirán dos planos. Uno fantasma, cuya raíz es el objeto eliminado y sus hijos son aquellos que lo eran antes de eliminar dicho objeto. El otro plano, corresponde al plano real cuya raíz es el objeto que reemplazó al eliminado y cuyos hijos son verdaderamente los más cercanos a la nueva raíz. Algunos de los objetos del plano fantasma no pertenecerán al plano real. Durante la inserción, los nuevos objetos se compararán con el centro real.

La elección del objeto de reemplazo resulta interesante, esto porque, dependiendo de la ubicación del elemento, el árbol puede resultar más o menos afectado. La alternativa general es elegir algún dato a partir de este nodo, de esta manera, sólo un subárbol es afectado y el resto del árbol permanece intacto y conserva absolutamente todas las propiedades de un VT. La figura 1 (b) representa el solapamiento de planos al eliminar el objeto S_1 de la figura 1 (a) al reemplazarlo por otro. Entonces llamaremos plano fantasma a $P(S_1)$ antes de su eliminación, el cual es representado en la misma figura por líneas no continuas.

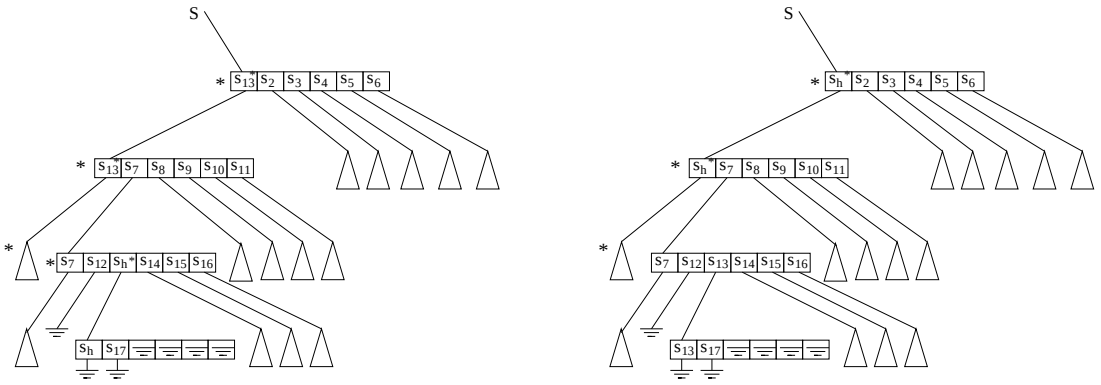
En [Uri05] se proponen cuatro alternativas de elección del objeto de reemplazo. Cada alternativa reemplaza en forma definitiva el objeto eliminado por otro, marcando el nuevo centro (subárbol) como afectado y conservando el radio cobertor original. La marca permite considerar los subárboles en forma especial en los procesos de eliminación y búsqueda. Las diferentes alternativas planteadas son:

Reemplazo por el descendiente más cercano : Una solución natural es elegir el elemento más cercano que sea *descendiente* del objeto eliminado, es decir, algún elemento dentro de su subárbol. Esta alternativa evita que el corrimiento del plano sea mayor. Sin embargo, la elección del más cercano a S_1 no garantiza que el plano siga igual, por lo tanto, es posible que varios puntos queden fuera del plano real formado por el nuevo dato y otros de planos adyacentes queden dentro (ver figura 1 (b)).

Como el proceso es recursivo, es muy probable que subplanos interiores sufran el mismo efecto, y por lo tanto, para una eliminación se crearían varios planos fantasmas dentro de un subárbol, además de los producidos por la replicación del proceso de eliminación.

Para visualizar como es afectada la estructura después de eliminar un centro, considere la figura 2, la cual representa un árbol que no contiene datos eliminados. Esta figura podría representar el plano indicado en la figura 1 (a) antes de eliminar objetos.

Si el centro eliminado es S_1, siendo el más cercano a él S_{15}, entonces la estructura quedaría según se



(a) reemplazo del más cercano

(b) reemplazo directo del último hijo ubicado en una hoja.

Figura 3: Voronoi-Tree: Árbol después de la eliminación de un centro usando planos fantasmas.

representa en la figura 3 (a), donde el asterisco representa la marca de *afectado*, tanto en el nodo como en el centro. El elemento S_h es el ubicado en una hoja que sería el último en moverse.
Es importante notar que dada la eliminación de un sólo elemento, el árbol se puede ver afectado en varios planos o subárboles interiores, esto dependiendo de la cantidad de niveles y la ubicación de los más cercanos.

Reemplazo por el más cercano en el nodo : Una modificación a esto, sería el reemplazo del más cercano, pero dentro de todos los subárboles que salen del nodo, de esta manera, la superposición de planos sería mínima, sin embargo, esto podría afectar además a subárboles adyacentes, lo que deformaría aún más el VT.

Para los dos métodos anteriores, hay que considerar los costos adicionales por búsqueda del más cercano, además, tomando en cuenta que si el objeto de reemplazo no está ubicado en una hoja, se realiza una nueva búsqueda del más cercano para este último objeto. Este proceso es recursivo hasta que uno de los objetos de reemplazo esté en una hoja. Este método implicaría que por cada eliminación existirían varios nodos y split marcados como afectados, además de las replicas, lo que afectaría en los costos a los métodos de eliminación y búsqueda.

Es importante señalar, que aunque un nodo se vea afectado, no necesariamente todos sus subárboles lo serán, lo que quiere decir que algunos de ellos siguen siendo VTs.

Reemplazo por Descendiente más Cercano Ubicado en una Hoja : Una tercera alternativa es el reemplazo por el objeto más cercano ubicado dentro de alguna hoja descendiente del elemento a borrar. Esto podría ocasionar un solapamiento mayor de los planos, pero con la garantía que se realiza sólo una búsqueda del más cercano y únicamente hay que recorrer el árbol hasta su primera hoja.

Reemplazo por descendiente hoja : La última alternativa es el reemplazo directo el elemento a borrar por el último descendiente que éste en alguna hoja del subárbol. Esta propuesta evita los cálculos de distancia ocasionados por la búsqueda del más cercano, es decir, costo 0 en términos de evaluaciones de distancia.

Las dos últimas alternativas tienen dos características importantes. La primera, es que sólo basta recorrer el árbol hasta su primera hoja. La segunda, que el movimiento del objeto no ocasiona overlap en los subárboles

interiores, por lo tanto, se puede decir que sólo un nodo y las replicas del centro borrado son afectados y no otros descendientes, es decir, el subárbol completo a partir del elemento borrado sigue siendo un VT.

El árbol de la figura 2 aplicando del último método, dejaría la estructura como se muestra en la figura 3 (b), considerando que la hoja donde se busca el reemplazante es la misma que para el primer algoritmo.

En [Uri05] se demuestra experimentalmente que la última alternativa se comporta mejor que tercera en la mayoría de los casos, salvo algunas excepciones en la eliminación. Durante la búsqueda, a bajos rangos y con porcentajes reducidos de datos eliminados la tercera alternativa tiene un comportamiento un poco mejor que la cuarta opción. Sin embargo, esta mejora sólo alcanza a un 1 % y decae a medida que aumenta el rango de búsqueda y la cantidad de datos eliminados, siendo finalmente superado por la última alternativa.

3.3. Búsquedas después de la eliminación

Para el algoritmo de *planos fantasmas*, el definir una marca de *centro afectado*, permite considerar a este centro en forma especial en los distintos métodos. La marca almacena la distancia entre el objeto eliminado y su reemplazante, lo que indicaría cuanto se movió el plano.

Es interesante mostrar que para el método de *planos fantasmas*, el algoritmo de inserción después de eliminaciones no sufre modificación, aunque los nuevos datos serían insertados en el plano real y no el fantasma, es decir, como hijo de un centro que realmente existe.

3.3.1. Búsquedas por rango y eliminaciones

Si se han realizado inserciones luego de eliminaciones, entonces la manera trivial de resolver la búsqueda por rango es incluir en el conjunto respuesta el subárbol cuyo centro está marcado, de esta manera se incluyen las posibles respuestas que estén tanto en el plano real como en el fantasma.

Si el grado los nodos del árbol es elevado y existiesen muchos centros marcados dentro de un nodo, entonces es posible estar incluyendo en las búsquedas más subárboles de los necesarios (todos los marcados).

Una optimización a lo anterior evita buscar en todos los marcados usando un factor de incertidumbre. Si se reemplaza S_1 por S_h, entonces $I = d(S_h, S_1)$, el cual es el valor almacenado en la marca de borrado. Por lo tanto, cualquier decisión respecto de ese subárbol puede estar erróneo por $\pm I$. Por ejemplo, durante la eliminación, al buscar un dato si $d(q, S_1) > d(q, S_2)$, se descarta S_1, ahora, si S_1 fue eliminado, entonces igualmente podemos descartar S_h si $d(q, S_h) > d(q, S_2) + I$.

Durante la búsqueda por rango se puede usar la misma optimización, por lo que se puede descartar S_h si $d(q, S_h) - rc(S_h) > r + I$, donde $rc(S_h) = rc(S_1) + I$.

Cabe recordar que si dentro de los subplanos (o subárboles) de dicho centro, existiese un nodo no marcado, entonces en este subplano no hay solapamiento, por lo tanto se aplica el método original y se mantiene la eficiencia dentro de este subárbol.

La segunda optimización propuesta para el egnat [Uri05] es una técnica propia de las estructuras basadas en pivotes. La idea es almacenar la distancia de un objeto a sólo un pivote. El pivote será el padre de dicho objeto y la distancia es calculada al momento de la inserción, por lo que no hay cálculos adicionales de distancia.

Esta optimización sería válida solamente para aquellos centros que nunca han sido padres, usualmente aquellos ubicados en las hojas. Esto debido a que dichos centros podrían estar marcados, por lo que la distancia con su padre no sería correcta, además, se utiliza el mismo espacio usado para almacenar el radio cobertor. Manteniendo la distancia al centro se puede usar la desigualdad triangular para evitar el cálculo directo de la distancia de un objeto consulta sobre los objetos que tienen las propiedades indicadas en el párrafo anterior. Esto es de la siguiente manera:

- Sea q el objeto consulta, p el centro raíz de un subárbol cualquiera, s_i los elementos hijos de p que no han sido padres, r el radio de búsqueda e I_p la marca de borrado insertada en p si este fuese un plano fantasma (I: distancia del objeto eliminado al objeto usado como reemplazo), entonces, si

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

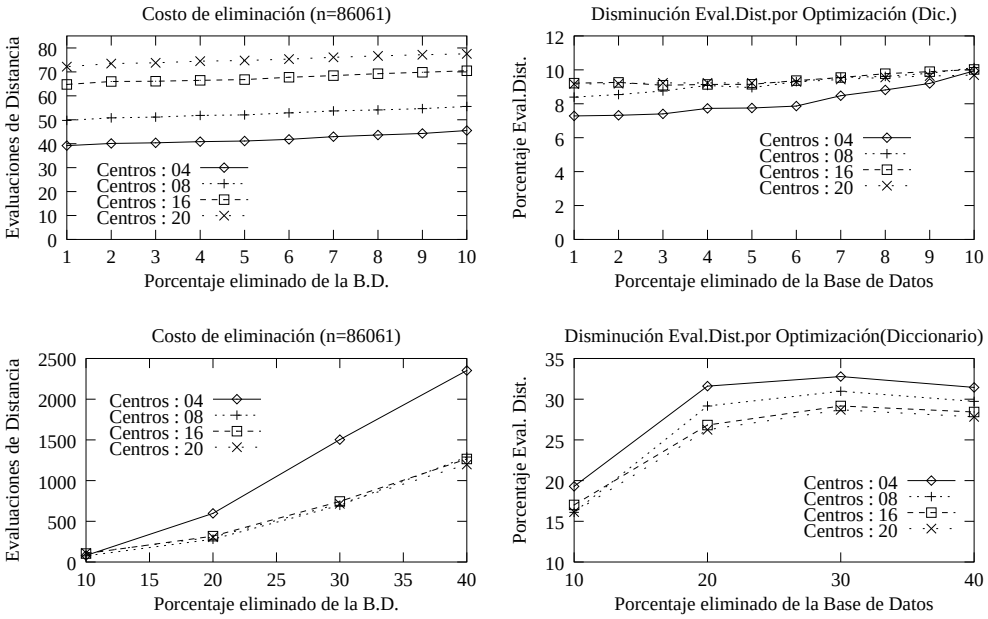


Figura 4: Costos de eliminación promedio para el diccionario Español y efectos de las optimizaciones respecto de la versión original (columna derecha).

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

4. Resultados experimentales

4.1. Eliminaciones

Para los experimentos se utilizaron dos espacios, el primero corresponde a un espacio de palabras, un diccionario en español de 86.061 objetos, donde la distancia utilizada es la *distancia de edición*. El segundo es un espacio de vectores de coordenadas reales de dimensión 10 generados con distribución de *Gauss* con media 1 y varianza 0.1, cuya cantidad de objetos es de 100.000, para este espacio se utilizó la *distancia Euclidiana*.

Para las eliminaciones, se construyó la estructura con el 90 % de los datos y sobre ella se eliminó el 10 o 40 % de los objetos, generados en forma aleatoria y en orden aleatorio.

Las figuras 4 y 5 muestran en la columna izquierda los costos de eliminaciones del 10 % y 40 % usando las optimizaciones, la columna derecha muestra los porcentajes de disminución respecto de la versión sin optimizaciones.

El método utilizado para la eliminación es el de *reemplazo por el último descendiente ubicado en una hoja*.

4.2. Búsquedas

Para los experimentos de búsquedas, se reservó un conjunto del 10 % de objetos, el cual es el que se buscará. Para el caso de los diccionarios, los rangos de búsqueda fueron 1, 2, 3 y 4. Para el caso de vectores, a priori se calculó el rango que recuperaban el 0.01, 0.1 y 1 por ciento de la base de datos. Para el caso de búsquedas con eliminaciones, se eliminó el 10 y 40 % de la base de datos y se reinsertó la misma cantidad, sobre esto se realizó la búsqueda del 10 % restante.

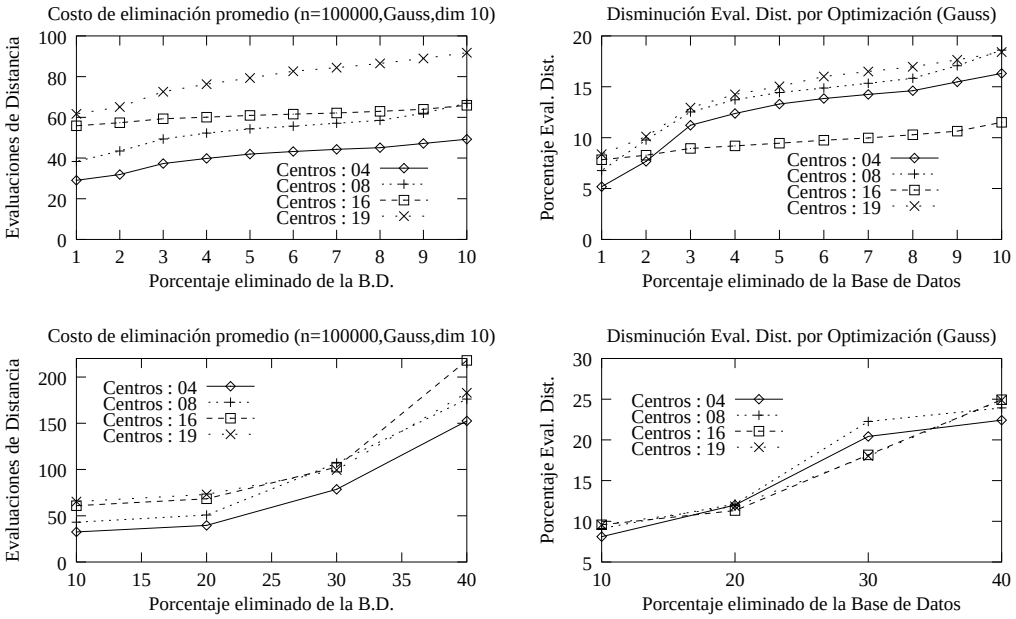


Figura 5: Costos de eliminación promedio para el espacio de Gauss y efectos de las optimizaciones respecto de la versión original (columna derecha).

La figuras 6 muestra los costos de búsqueda sobre la estructura sin eliminaciones, con eliminación e inserción de un 10 % de datos y eliminación e inserción de un 40 %. La columna derecha muestra los efectos de la optimización y el porcentaje de reducción de evaluaciones de distancia respecto de la versión original. El método de búsqueda incluye las optimizaciones. Respecto del espacio de palabras, la aplicación de las optimizaciones implican mejoras sobre el 50 % en el mejor caso, para 20 centros y sin eliminaciones; en el peor caso, la estructura con 4 centros y con 40 % de objetos eliminados y reinsertados, las optimizaciones implicaron mejoras de 16 % aproximadamente.

5. Conclusiones

Se ha presentado una versión dinámica de la estructura *Voronoi-Tree*, la cual permite realizar inserciones y eliminaciones eficientemente sin afectar la calidad de las búsquedas.

Se ha demostrado experimentalmente la eficiencia del método de *planos fantasmas* y las optimizaciones sobre una estructura de tipo árbol.

Es importante destacar que la optimización que modifica la estructura y mantiene la distancia al padre (pivote), logra reducciones de entre 17 % y 35 % para la búsqueda por rango en la estructura sin eliminaciones. En el espacio de palabras logra mejoras del orden del 50 %.

Respecto de los experimentos, se puede concluir, como era de esperar, que la búsqueda se degrada a aumentar el porcentaje de objetos eliminados, sin embargo, el aumento en las evaluaciones de distancia respecto de la estructura sin eliminaciones sigue siendo adecuado.

Las alternativas propuestas también han sido inicialmente las más eficientes en el trabajo que se realiza en la actualidad, que es proveer de una estructura dinámica eficiente que trabaje en memoria secundaria, aquí también son relevantes junto con las evaluaciones de distancia, los accesos a disco y el tamaño de la estructura en disco.

Algunas extensiones futuras al presente trabajo implican encontrar mecanismos de elección del objeto de reemplazo para disminuir los costos de búsquedas. También se propone analizar y experimentar el método de

planos fantasmas sobre otros árboles como también sobre otros tipos de estructuras métricas, como ser las del tipo arreglos. Finalmente, proyectar la estructura a memoria secundaria y paralelizar los algoritmos.

Referencias

[BK73] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Communication of ACM*, 16(4):230–236, 1973.

[Bri95] Sergei Brin. Near neighbor search in large metric spaces. In *the 21st VLDB Conference*, pages 574–584. Morgan Kaufmann Publishers, 1995.

[BYCMW94] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixedqueries trees. In *5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.

[CNBYM01] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José L. Marroquín. Searching in metric spaces. In *ACM Computing Surveys*, pages 33(3):273–321, September 2001.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree : An efficient access method for similarity search in metric spaces. In *the 23st International Conference on VLDB*, pages 426–435, 1997.

[DN87] F. Dehne and H. Noltemeier. Voronoi trees and clustering problems. *Informations Systems*, 12(2):171–175, 1987.

[KM83] I. Kalantari and G. McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9(5), 1983.

[Nav02] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, 11(1):28–46, 2002.

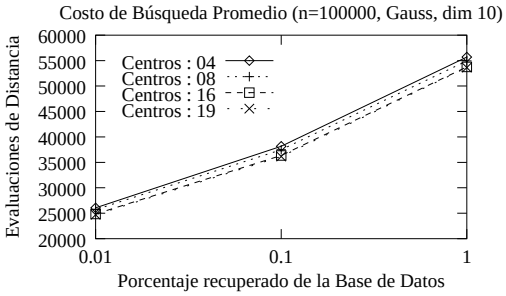
[NR02] Gonzalo Navarro and Nora Reyes. Fully dynamic spatial approximation trees. In *the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, pages 254–270, Springer 2002.

[Uhl91] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. In *Information Processing Letters*, pages 40:175–179, 1991.

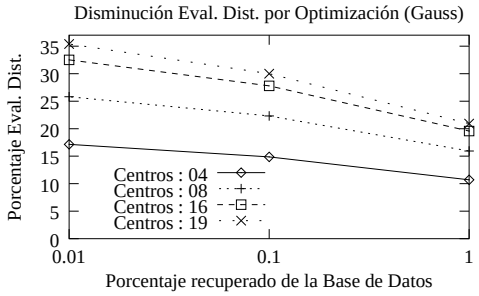
[UN03] R. Uribe and G. Navarro. Una estructura dinámica para búsqueda en espacios métricos. In *Actas de las XI Jornadas Chilenas de Computación*, Chillán, Chile, 2003. In Spanish. In CD-ROM.

[Uri05] Roberto Uribe. Manipulación de estructuras métricas en memoria secundaria. Master's thesis, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile, Abril 2005.

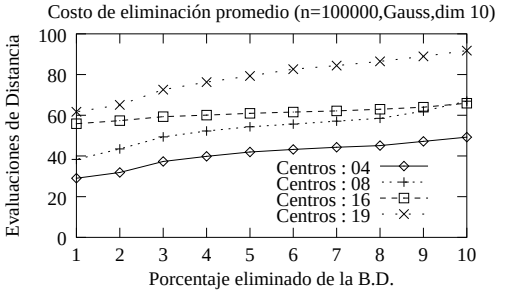
[Yia93] P. Yianilos. Data structures and algoritms for nearest neighbor search in general metric spaces. In *4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, 1993.



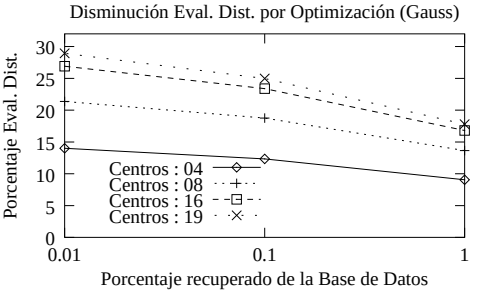
(a) Sin eliminaciones



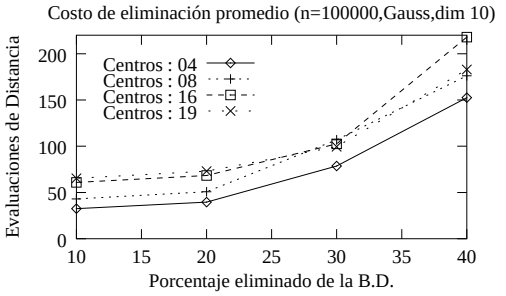
(b) Reducción por optimización



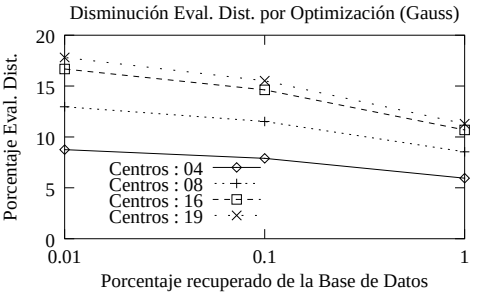
(c) 10 % eliminado y 10 % reinsertado



(d) Reducción por optimización



(e) 40 % eliminado y 40 % reinsertado



(f) Reducción por optimización

Figura 6: Costos de búsqueda promedio para el espacio de Gauss. Sin eliminaciones y con 10 % y 40 % de datos eliminados y reinsertados.