Consultas Métrico Espaciales

Adrián Planas[1] , Andrés Pascal[1], Norma Herrera[2]

[1] Univ. Tec. Nacional, Fac. Concepción del Uruguay, Dpto. Sistemas de Información
pladnic@gmail.com, andrespascal2003@yahoo.com.ar

[2] Univ. Nac. de San Luis, Departamento de Informática, San Luis, Argentina
nherrera@unsl.edu.ar

Abstract. Los espacios métricos permiten modelar bases de datos que soportan búsquedas por similitud, es decir, búsquedas de objetos parecidos a uno dado. Las bases de datos espaciales permiten almacenar y recuperar eficientemente datos que poseen alguna componente espacial. Existen aplicaciones donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también el componente espacial. Este tipo de consultas no puede resolverse eficientemente ni con índices espaciales, ni con índices métricos. En este artículo abordamos el estudio de estas consultas con el fin de formalizarlas y proponer un método eficiente para su resolución. Para ello, presentamos el MeTree, una combinación del índice métrico FQT con el índice espacial R-Tree que permite consultas combinando ambos aspectos.

Keywords: Consultas Métrico-Espaciales, Búsquedas por Similitud, Índices Métricos, Índices Espaciales.

1 Introducción

En las últimas tres décadas, las bases de datos han incorporado paulatinamente la capacidad de almacenar tipos de datos no estructurados tales como imágenes, sonido, texto, video, elementos geométricos, etc. La problemática de almacenamiento y búsqueda en éstas difiere notablemente de la correspondiente a las bases de datos clásicas. Estos tipos de datos no poseen una longitud fija, por lo cual son más difíciles de organizar dentro de una estructura, normalmente no pueden ser ordenados, y la búsqueda exacta carece de interés en este ámbito. Es en este contexto donde han surgido nuevos modelos de bases de datos capaces de cubrir eficaz y eficientemente las necesidades de almacenamiento y búsqueda de estas aplicaciones. Entre estos modelos se encuentran las bases de datos espaciales, que permiten mantener un registro de ubicación de un objeto en un espacio n-dimensional, y los espacios métricos, que constituyen un modelo genérico que permite las búsquedas por similitud.

El objetivo de las búsquedas por similitud [1] es encontrar objetos que poseen características similares a uno dado bajo cierto criterio. Esta clase de funcionalidad ha sido utilizada en muchas áreas de la informática. Por ejemplo, en reconocimiento de patrones, las consultas por similitud se pueden usar para clasificar nuevos objetos de acuerdo a objetos cercanos ya clasificados; en sistemas de recomendación se pueden utilizar para generar recomendaciones personalizadas basadas en las preferencias del usuario; en bases de datos de imágenes se pueden usar para búsquedas por contenido.

Dado que los tipos de datos de estos sistemas pueden variar considerablemente (imágenes, cadenas, secuencias de proteínas, sonido, texto libre, etc), es deseable contar con un modelo genérico que los abarque, y que permita diferentes funciones de comparación. En este aspecto, el modelo más utilizado es el definido por los Espacios Métricos, que permiten cualquier función de similitud, siempre que cumpla con ciertas propiedades.

Un espacio métrico es un par (U, d) donde U es un universo de objetos y $d: UxU \rightarrow R+$ es una función de distancia definida entre los elementos de U que mide la similitud (disimilitud, en realidad) entre ellos; esto significa que a menor distancia, más cercanos o similares son los objetos. Esta función d cumple con las propiedades características de una función métrica:

(a) $\forall x,y \in U$, $d(x, y) \geq 0$ (positividad)

(b) $\forall x,y \in U$, $d(x, y)=d(y, x)$ (simetría)

(c) $\forall x \in U$, $d(x, x)= 0$ (reflexividad)

(d) $\forall x,y,z \in U$, $d(x, y) \leq d(x, z) + d(z, y)$ (desigualdad triangular)

Esta última propiedad, la desigualdad triangular, es de suma importancia para descartar elementos durante la búsqueda. La base de datos será entonces un subconjunto finito $X \in U$ de cardinalidad n. Una de las consultas típicas en este modelo es la búsqueda por rango, que se denota $(q, r)_d$. Dado un elemento $q \in U$ al que llamaremos *query*, y un radio de tolerancia r, una búsqueda por rango consiste en recuperar los objetos de X que estén a distancia a lo sumo r de q, es decir:

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

Por otro lado, el modelo espacial ha aumentado su uso significativamente desde la década de los 90', principalmente como soporte para los Sistemas de Información Geográfica (SIG). Con el crecimiento de Internet, la cantidad de objetos que poseen una ubicación espacial se ha incrementado exponencialmente. Por ejemplo, los dispositivos móviles han permitido la generación de cantidades enormes de datos georeferenciados que se comparten a través de la Web; principalmente fotos con ubicación.

Las bases de datos espaciales están formadas por información estructurada a la que se le añaden objetos geométricos: puntos, polilíneas o polígonos, bajo un sistema de referencia predefinido. Dichas bases pueden contener cientos de miles o millones de estos objetos sobre los cuales se realizan consultas con operaciones tales como intersección, adyacencia, inclusión y muchas otras. Tales operaciones suelen ser costosas desde el punto de vista computacional, por lo cual se requieren métodos de acceso que disminuyan considerablemente la cantidad de comparaciones necesarias para resolver una consulta.

Existen aplicaciones donde resulta de interés realizar búsquedas por similitud pero teniendo en cuenta también la componente espacial. Un caso típico son los SIG, donde los elementos con ubicación espacial en muchos casos requieren ser consultados también por similitud. Por ejemplo, en un mapa con fotos de edificios y construcciones ampliamente conocidas, sería de interés, dada una imagen de una construcción, encontrar las imágenes similares dentro de un área geográfica particular. O en lugar de una imagen, simplemente buscar por similitud el nombre de un punto de interés más cercano a la ubicación donde uno se encuentra, ya que en ocasiones no se conoce cómo se escribe. Estas consultas implican una búsqueda que tiene en cuenta tanto el aspecto espacial como la similitud del elemento que se consulta y, en el caso de segundo ejemplo, son muy comunes cuando se utilizan Sistemas de Posicionamiento Global (GPS). Un caso particular de este problema, la búsqueda de documentos geográficos, ha sido estudiado previamente dando lugar a índices tales como el IR-Tree [2], pero que están diseñadas sólo para documentos.

Si bien estas consultas se pueden resolver utilizando índices espaciales y métricos por separado para luego encontrar la respuesta final, es mucho más eficiente contar con métodos de acceso que estén diseñados específicamente para resolverlas. En este artículo se presenta

una solución general a este problema, que sirve para cualquier objeto que sea consultado por similitud y ubicación a la vez.

Este artículo está organizado de la siguiente manera. En la Sección 2 se presenta un breve resumen del trabajo relacionado. En las Secciones 3 y 4 planteamos el problema de consultas métrico-espaciales y proponemos un primer índice para resolverlas con eficiencia, el MeTree. En la Sección 5 se muestra su evaluación experimental, y por último en la Sección 6 se presentan las conclusiones y el trabajo futuro.

2 Trabajo Relacionado

En esta sección se resumen los desarrollos más importantes en cuanto a los métodos de acceso métricos y espaciales.

2.1 Índices Métricos

Existen varios métodos de acceso diseñados para acelerar la búsqueda por similitud en espacios métricos genéricos. En general, pueden ser clasificados en dos categorías: métodos basados en particiones compactas [3], [4], [5], [6] y métodos basados en pivotes [7], [8], [9], [10]. Los métodos basados en particiones compactas dividen el espacio en regiones representadas por centros e intentan descartar las regiones alejadas del objeto que se consulta. Mientras que los basados en pivotes almacenan las distancias precalculadas de cada objeto hacia los pivotes y las utilizan con el mismo de fin. En ambos casos se utiliza la desigualdad triangular para reducir la cantidad de elementos a comparar con la consulta. En general, los métodos basados en pivotes obtienen mejores resultados en cuanto a la performance de las consultas.

Dentro de los basados en particiones compactas, el BST [11], [12] es un árbol binario construido recursivamente. Utiliza un centro con un radio de cobertura para representar cada partición. El GHT [13] utiliza dos centros por cada nodo del árbol y agrupa los elementos de acuerdo al centro más cercano a cada uno de ellos. GANT [14] es una generalización del GHT. Utiliza particiones de Voronoi del espacio. Existe una versión dinámica de este último, llamada EGANT [15]. SAT [16] utiliza un modelo complementario a los diagramas de Voronoi, los grafos de Delaunay. También existen extensiones de SAT, dinámicas, y para memoria secundaria [17], [18]. El M-tree [4] es un árbol balanceado optimizado para memoria secundaria, que surge como una adaptación natural de la familia de los B-Trees a más de una dimensión. Existen muchas variantes del M-Tree, tales como Slim-Tree [6], DBM-Tree [19] y CM-Tree [20]. El D-index [5] es una estructura que utiliza una función hash para mapear objetos en buckets. LC [3] utiliza una lista de clusters que mejora la eficiencia en la búsqueda a costa de hacer menos eficiente su construcción. BP [21] es un árbol no-balanceado pensado para espacios métricos de alta dimensionalidad.

Respecto a los métodos basados en pivotes, AESA [22] emplea una tabla que registra todas las distancias entre los objetos de la base de datos. Para reducir el tamaño de dicha tabla, se han propuesto distintas variantes. Por ejemplo, LAESA [9] solo guarda las distancias hacia un conjunto de pivotes seleccionados. EP [23] selecciona un conjunto de pivotes sin redundancia, que cubren la base de datos completa. Clustered Pivot-Table [24] agrupa las distancias precalculadas para mejorar aún más la eficiencia de las búsquedas. El BKT [25] fue uno de los primeros índices métricos basados en pivotes, y está diseñado para distancias discretas. Es un árbol donde cada nodo contiene un pivote distinto y todos los elementos que

se encuentran a la misma distancia de pivote se ubican en el mismo nodo hijo. El FQT [26] es similar al BKT, pero utiliza el mismo pivote para todos los nodos del mismo nivel, reduciendo así la cantidad de comparaciones entre la consulta y los pivotes. VPT [10] es un árbol binario diseñado para distancias continuas y su versión r-aria es el MVPT [27].

Existen también métodos híbridos, que combinan particiones compactas con pivotes, por ejemplo el PM-Tree [28], que utiliza pivotes sobre regiones definidas por un M-Tree, o el M-Index [29], que agrupa objetos utilizando distancias precalculadas hacia sus pivotes más cercanos.

2.1 Índices Espaciales

El procesamiento de consultas espaciales implica la ejecución de operaciones geométricas complejas y costosas. Considerando que las bases de datos espaciales suelen contener grandes cantidades de objetos geométricos, realizar un recorrido secuencial para resolver una consulta espacial no es una solución práctica en la mayoría de los casos, por lo cual en las aplicaciones reales es necesario el uso de índices espaciales.

Los métodos de acceso espacial también pueden clasificarse en dos categorías [30]: estructuras dirigidas por el espacio, y estructuras dirigidas por los datos. Las primeras están basadas en la partición de un espacio 2D en regiones rectangulares. Los objetos se mapean en las regiones de acuerdo a algún criterio geométrico. En el segundo caso las particiones están basadas en la distribución del conjunto de objetos que se está indexando.

El Grid File [31] divide el espacio en celdas fijas para indexar puntos. Cada celda está asociada a una página de disco, donde se almacenan secuencialmente los objetos contenidos en dicha celda. Si la celda se llena, la celda se divide en dos. Existen variantes del mismo para indexar rectángulos. El QuadTree [32] subdivide el plano en cuatro cuadrantes del mismo tamaño, y realiza esta misma operación sobre cada cuadrante, dando como resultado un árbol 4-ario. Cada nodo representa a todos los objetos contenidos en su cuadrante. El QuadTree ha sido utilizado ampliamente tanto para la búsqueda espacial como para distintas tareas de procesamiento de imágenes. El Bitplane Quadtree (BQ-Tree) [33] es una variante diseñada para datos geoespaciales de gran escala. El K-D-Tree [34] almacena puntos k-dimensionales subdividiendo el espacio en forma alternativa en paralelepípedos rectángulos ortogonales a cada eje de coordenadas. Existe una familia de métodos llamada “Space Filling Curve”, que se basan en funciones que transforman un espacio n-dimensional en una sola dimensión, conservando ciertas propiedades de proximidad. Un ejemplo es el Hilbert Space Filling Curve [35].

Respecto a los métodos dirigidos por los datos, los más importantes son la familia de índices R-Tree [36], que constituyen una generalización del B-Tree a más de una dimensión. Son árboles M-arios de rectángulos donde cada nodo contiene espacialmente a todos los elementos del subárbol del que es raíz. El árbol es balanceado y cada nodo interior salvo la raíz, contiene entre m y M hijos, y se almacena en una página de disco. Son los índices espaciales más utilizados actualmente por los motores de bases de datos comerciales.

3 Modelo Métrico-Espacial

Las aplicaciones donde tienen sentido las consultas métrico-espaciales tienen las siguientes características:

- No se pueden realizar búsquedas exactas sobre los objetos: los elementos de la base de datos no tienen un identificador (o un grupo de atributos) que se pueda utilizar como clave de búsqueda, o si existe, no se conoce en el momento de la consulta.
- Los objetos tienen una ubicación (y/o forma) espacial.
- Los resultados de una consulta tienen que satisfacer requisitos tanto de similitud como de posición espacial.
- La base de datos contiene una cantidad suficientemente grande de objetos o el tiempo de respuesta ante una consulta debe ser suficientemente reducido como para que no tenga sentido realizar una búsqueda secuencial.

Sea U el universo de objetos válidos, el modelo Métrico-Espacial se define mediante el par (U, d), donde para todo $o \in U$, la función $s(o) \in S$ devuelve el componente métrico del objeto y $e(o) \in E$ su aspecto espacial. La función métrica d, es la medida de disimilitud y está definida como $d:SxS \rightarrow R+$. Por ejemplo, si $o \in U$ es un punto de interés en un mapa, $s(o)$ podría ser una cadena que representa su nombre o una foto de su frente, mientras que $e(o)$ será el punto que indica su ubicación.

Una consulta por rango métrico e intersección espacial, se denota a través de la 3-upla $(q, r, g)_d$ y se define formalmente de la siguiente manera:

$$(q, r, g)_d = \{o \in X / d(s(o), q) \leq r \wedge intersects(e(o), g)\}$$

siendo $X \subseteq U$, la base de datos, q el aspecto métrico de la consulta, r el radio de búsqueda que representa el valor máximo de disimilitud aceptada, y g su aspecto geométrico (punto, polilínea o polígono).

Una forma trivial de resolver una consulta métrico-espacial es el uso de dos índices, uno métrico y el otro espacial. Luego, ante una consulta $(q, r, g)_d$, se procede de la siguiente manera:

1. Realizar la búsqueda por similitud $(q, r)d$ sobre el índice métrico, devolviendo el conjunto L como resultado,
2. Realizar una búsqueda de los elementos que se intersectan con g, sobre el índice espacial, devolviendo como resultado el conjunto M,
3. Por último, realizar la intersección $L \cap M$ para obtener el resultado final.

La desventaja de esta solución es que no aprovecha la información métrica y espacial al mismo tiempo para el descarte de elementos. Una mejor estrategia es el diseño de un índice que integre ambos aspectos y permita consultas métricas, espaciales y métrico-espaciales.

4 Método de Acceso Métrico-Espacial: el MeTree

En este trabajo se presenta un primer índice métrico-espacial, el MeTree, diseñado en base a una variante del índice métrico FQT (que permite distancias continuas) y a la familia de índices espaciales R-Tree. El MeTree es un árbol r-ario en el cual cada nodo contiene un rectángulo que contiene espacialmente a todos sus hijos, tal como el R-Tree. Pero además, cada nodo tiene un intervalo métrico que representa el valor mínimo y máximo de las distancias de todos los elementos del subárbol hacia un pivote correspondiente al nivel del nodo (como un FQT). La altura del árbol es fija en principio, y está determinada por la cantidad de pivotes con la cual se decide indexar. Sin embargo, esta cantidad puede ser extendida en cualquier momento, agregando nuevos pivotes y por lo tanto, nuevos niveles. Una hoja del árbol contendrá apuntadores a objetos que sean similares y que además, estén cercanos espacialmente.

Un problema importante es que puede haber elementos similares muy alejados espacialmente. En los niveles inferiores del árbol esta situación produciría rectángulos demasiado grandes, afectando considerablemente a la eficiencia del índice. Para resolver este problema se establecieron restricciones a cumplir para que dos elementos puedan pertenecer a la misma hoja, y se decidió permitir que los intervalos métricos de nodos hermanos no sean excluyentes, como así tampoco los rectángulos.

Un nodo interior del MeTree es una 3-upla *(i_m, f_m, rect)*, donde i_m.f_m es el intervalo métrico y *rect* el rectángulo que contiene espacialmente a todos los hijos de dicho nodo. A su vez, cada nivel del árbol (salvo la raíz) tiene asociado un pivote.

En la Fig. 1 se presenta un ejemplo de la estructura de un MeTree. En la parte superior se muestran los nodos del árbol y los intervalos métricos de cada nodo, de acuerdo las distancias de los elementos al pivote de cada nivel. Por razones de legibilidad, los rectángulos de cada nodo se muestran por separado, en la parte inferior de la figura. Como se puede ver, puede haber superposición tanto entre intervalos métricos de los nodos hermanos, como en los rectángulos que representan el espacio que ocupan.

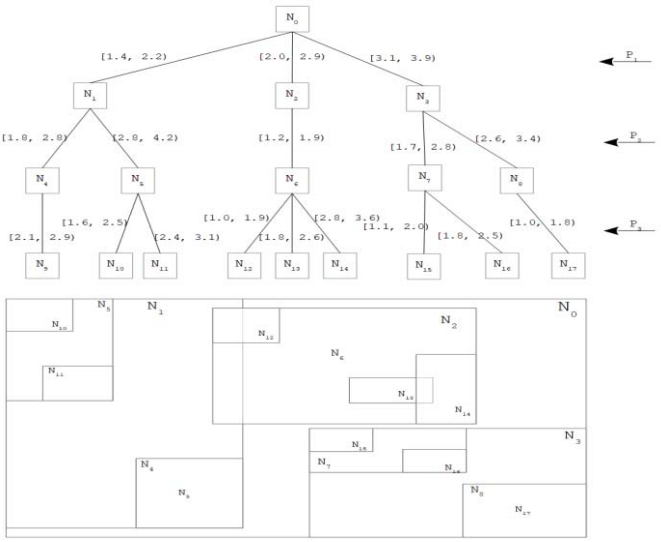

Fig. 1. Estructura de un MeTree.

Ante una inserción se procede de la siguiente manera. Sea *o* el objeto a insertar y p_n el pivote correspondiente al nivel *n*, se calcula el costo de agregar el objeto en cada uno de los nodos hijos del nodo actual y se obtiene el mínimo. Si dicho costo es menor que *infinito*, el objeto se añade al hijo de costo mínimo, y en caso contrario se agrega un nuevo hijo. El costo se calcula como la suma ponderada del costo métrico y el costo espacial. El costo métrico es cero si $i_m \leq d(s(o),p_n) \leq f_m$, es decir, si la distancia del objeto al pivote del nivel se encuentra dentro del intervalo métrico del nodo. En caso contrario es igual al menor incremento del intervalo, necesario para incluir a $d(s(o), p_n)$. Si el intervalo aumentado es mayor que el tamaño máximo permitido, el costo es *infinito*. Un procedimiento similar se utiliza para calcular el costo espacial. Si el aspecto espacial del objeto a insertar se encuentra ubicado dentro del rectángulo del nodo, el costo es cero. Para calcular el costo de aumentar el rectángulo para incluir al objeto se calcula el incremento necesario de su diagonal, y si es mayor a un valor máximo, nuevamente se considera *infinito*. Cuando se añade un elemento a un nodo existente, tanto su intervalo métrico como su rectángulo pueden aumentar de

tamaño. El tamaño máximo de los intervalos métricos es un parámetro fijado de antemano. En el caso de los rectángulos, la diagonal máxima depende del nivel del nodo. La raíz contiene el espacio completo donde pueden estar los elementos. En cada nivel la diagonal máxima es la mitad de la diagonal del nivel padre. Es decir que los tamaños disminuyen logarítmicamente.

Cuando se realiza una consulta métrico-espacial $(q, r, g)_d$, se visitan los nodos que cumplen las condiciones $im\text{-}r \leq d(q, p_n) \leq fm+r$ y *intersects(g, rect)* hasta alcanzar las hojas, obteniendo un conjunto de candidatos. Luego se recorre secuencialmente este conjunto y se comparan los elementos con la consulta para obtener el resultado final. En la Fig. 2 se muestra el pseudocódigo de la consulta.

```
ConsultaMeTree(q, r, g)
  nodosAVisitar:=hijosDe(raíz) // conjunto de hijos
  nivelActual:=1
  WHILE |nodosAVisitar|>0 AND nivelActual <=MaxNiv:
    nodosNuevoNivel:=[]  // conjunto vacío
    dist:=distanciaMetrica(q, pivote(nivelActual))
    FOR padre IN nodosAVisitar:
      FOR hijo IN hijosDe(padre):
        IF (dist BETWEEN hijo.im-r AND hijo.fm+r):
          IF intersects(g, hijo.rect):
            nodosNuevoNivel += hijo  // agregar hijo
    nodosAVisitar:= nodosNuevoNivel
    nivelActual += 1
  // en nodosAVisitar quedan las hojas de candidatos
  Resultado:=[]
  FOR hoja IN nodosAVisitar:
    elementos:=elementosDe(hoja)
    FOR elem IN elementos:
      dist:= distanciaMetrica(q, s(elem))
      IF (dist<=r) AND (intersects(g, e(elem))):
        Resultado += e
  Return Resultado
```

Fig. 2. Pseudocódigo de Consulta a un MeTree.

Este mecanismo aprovecha tanto el aspecto métrico como el espacial para descartar ramas del árbol, lo que aumenta la eficiencia significativamente en comparación a la solución trivial.

5 Resultados Experimentales

Como parte de este trabajo se implementó el MeTree y se realizaron pruebas preliminares sobre una base de datos de 1514 puntos de interés correspondientes al microcentro de la Ciudad Autónoma de Buenos Aires (Fig. 3). Las consultas se realizaron sobre los nombres de los puntos de interés, utilizando la distancia de Levenshtein como función métrica, y su ubicación geográfica.

Se definieron 100 consultas compuestas por nombres similares a los elementos de la base de datos, radios aleatorios y polígonos de distintos tamaños y formas. Se eligieron aleatoriamente elementos de la base de datos para utilizarlos como pivotes y se fijó el tamaño máximo de los intervalos métricos.

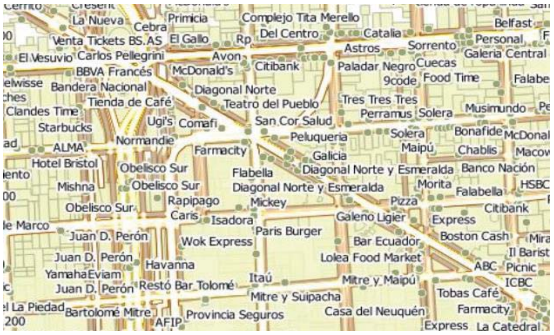

Fig. 3. Sección del mapa utilizado en los experimentos.

Las consultas se ejecutaron utilizando la solución trivial planteada anteriormente y el MeTree y se compararon sus resultados. Ya que tanto la función de distancia métrica como la operación de intersección de dos elementos geométricos suelen ser costosas, se tomó como medida de comparación la cantidad de elementos candidatos resultantes sobre los cuales se hace la comprobación secuencial de ambos aspectos. Los resultados obtenidos para cada una de las consultas se muestran en la Fig. 4.

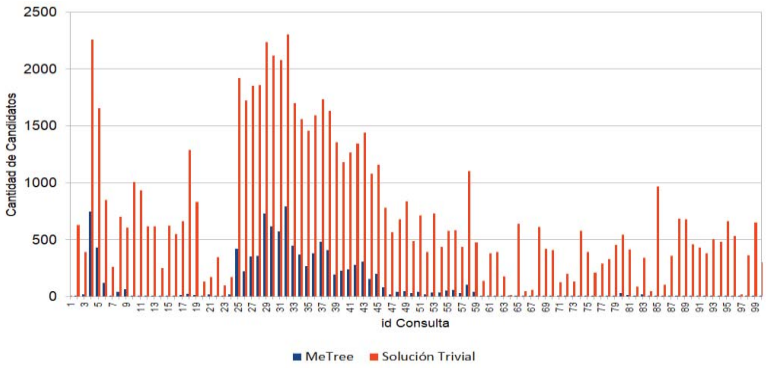

Fig. 4. Costos del MeTree y de la Solución Trivial.

En promedio, la cantidad de candidatos por consulta, que luego deben ser comparados con la consulta para determinar si forman parte del resultado final o no, fue 104,24 para el MeTree y 740,09 para la solución trivial. La diferencia en eficiencia es significativa y se explica por el hecho de que el descarte mediante ambos aspectos a la vez reduce considerablemente la cantidad de elementos a evaluar durante el recorrido del árbol.

Si bien los resultados son promisorios, estos experimentos son los primeros de una serie de pruebas planificadas que brindarán mayor detalle y confianza en el funcionamiento del índice bajo distintas situaciones y con mayores cantidades de datos.

6 Conclusiones y Trabajo Futuro

En este trabajo formalizamos el modelo métrico-espacial y presentamos el método de acceso MeTree, que resuelve consultas con restricciones por similitud y espaciales al mismo tiempo. Trabajos de investigación anteriores limitaban las búsquedas por similitud y espacial sólo a

búsquedas de documentos. El índice presentado en este artículo permite trabajar con cualquier tipo de objeto, siempre que exista una distancia métrica de comparación.

Un aspecto interesante es que el mismo índice se puede utilizar para realizar búsquedas por similitud o búsquedas espaciales por separado también, es decir que en principio no sería necesario contar con índices espaciales y métricos adicionales. Pero para comprobar esta afirmación aún se requiere verificación de los resultados bajo distintas situaciones y mayores cantidades de datos, ya que los experimentos realizados aún no son suficientes.

Trabajo futuro

1. Calcular la complejidad temporal del método en forma analítica, tema en el cual estamos trabajando actualmente.
2. Realizar experimentos con otros tipos de datos (imágenes, por ejemplo), con funciones de distancia de mayor complejidad y mayor cantidad de datos (estamos preparando una base de más de un millón de elementos)
3. Analizar y realizar modificaciones al procedimiento de inserción, para disminuir el tamaño de los rectángulos y de los intervalos métricos y reducir la superposición.

Referencias

1. Chávez, E., Navarro, G., Baeza-Yates, R. and Marroquín, J.L. Searching in metric spaces. ACM Computing Surveys, 33(3):273.321, September (2001)
2. LI, Zhisheng; LEE, Ken C. K.; ZHENG, Baihua; LEE, Wang-Chien; LEE, Dik Lun; and WANG, Xufa. IR-Tree: An Efficient Index for Geographic Document Search. IEEE Transactions on Knowledge and Data Engineering. (2011)
3. E. Chavez and G. Navarro, "A compact space decomposition for effective metric indexing," Pattern Recognition Letters, 26(9), pp. 1363–1376, (2005)
4. P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in VLDB, pp. 426–435. (1997)
5. V. Dohnal, C. Gennaro, P. Savino, and P. Zezula, "D-index: Distance searching index for metric data sets," Multimedia Tools Appl., 21(1), pp. 9–33, (2003)
6. C. T. Jr., A. Traina, B. Seeger, and C. Faloutsos, "Slim-trees: High performance metric trees minimizing overlap between nodes," in ICDE, pp. 51–65. (2000)
7. W. Burkhard and R. Keller, "Some approaches to best-match file searching," Commun. ACM, 16(4), pp. 230–236, (1973)
8. C. T. Jr., R. F. S. Filho, A. J. M. Traina, M. R. Vieira, and C. Faloutsos, "The Omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient," VLDB J., 16(4), pp. 483–505, (2007)
9. L. Mico, J. Oncina, and R. C. Carrasco, "A fast branch & bound nearest neighbour classifier in metric spaces," Pattern Recognition Letters, 17(7), pp. 731–739, (1996)
10. P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in SODA, pp. 311-321.(1993)
11. I. Kalantari and G. McDonald, "A data structure and an algorithm for the nearest point problem," IEEE Trans. Software Eng., 9(5), pp. 631–634, (1983)
12. H. Noltemeier, K. Verbarg, and C. Zirkelbach, "Monotonous bisector Trees — A tool for efficient partitioning of complex scenes of geometric objects," in Data Structures and Efficient Algorithms, pp. 186–203, (1992)
13. J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," Inf. Process. Lett., 40(4), pp. 175–179, (1991)
14. S. Brin, "Near neighbor search in large metric spaces," in VLDB, pp. 574–584, (1995)
15. G. Navarro and R. U. Paredes, "Fully dynamic metric access methods based on hyperplane partitioning," Inf. Syst., 36(4), pp. 734–747, (2011)

16. G. Navarro, “Searching in metric spaces by spatial approximation,” VLDB J., 11(1), pp. 28–46, (2002)
17. L. Britos, A. M. Printista, and Nora Reye, “DSACL+-tree: A dynamic data structure for similarity search in secondary memory,” in SISAP, pp. 116–131, (2012)
18. G. Navarro and N. Reyes, “Dynamic spatial approximation trees for massive data,” in SISAP, pp. 81–88, (2009)
19. M. R. Vieira, C. T. Jr., F. J. T. Chino, and A. J. M. Traina, “DBM-Tree: A dynamic metric access method sensitive to local density data,” J. Inf. Data Management, 1(1), pp. 111–128, (2010)
20. L. Aronovich and I. Spiegler, “CM-tree: A dynamic clustered index for similarity search in metric databases,” Data Knowl. Eng., 63(3), pp. 919–946, (2007)
21. J. Almeida, R. D. S. Torres, and N. J. Leite, “BP-tree: An efficient index for similarity search in high-dimensional metric spaces,” in CIKM, pp. 1365–1368, (2010)
22. E. Vidal, “An algorithm for finding nearest neighbors in (approximately) constant average time,” Pattern Recognition Letters, 4(3), pp. 145–157, (1986)
23. G. Ruiz, F. Santoyo, E. Chavez, K. Figueroa, and E. S. Tellez, “Extreme pivots for faster metric indexes,” in SISAP, pp. 115–126. (2013)
24. J. Mosko, J. Lokoc, and T. Skopal, “Clustered pivot tables for I/O optimized similarity search,” in SISAP, pp. 17–24 (2011)
25. W. Burkhard and R. Keller, “Some approaches to best-match file searching,” Commun. ACM, 16(4), pp. 230–236, (1973)
26. R. A. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. “Proximity matching using fixed-queries trees,” in CPM, pp. 198–212, SODA, (1994)
27. T. Bozkaya and M. Ozsoyoglu, “Distance-based indexing for high dimensional metric spaces,” in SIGMOD, pp. 357–368, (1997)
28. T. Skopal, J. Pokorny, and V. Snasel, “PM-tree: Pivoting metric tree for similarity search in multimedia databases,” in ADBIS, pp. 803–815, (2004)
29. D. Novak, M. Batko, and P. Zezula, “Metric Index: An efficient and scalable solution for precise and approximate similarity search,” Inf. Syst., 36(4), pp. 721–733, (2011)
30. Rigaux P, Scholl M, Voisard A. “6 - Spatial Access Methods”. En Spatial Databases, Morgan Kaufmann, San Francisco, pp 201–266, (2002)
31. Nievergelt J, Hinterberger H, Sevcik K.C. The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Trans. Database Syst 9(1):38–71, (1984)
32. Finkel, R. A.; Bentley, J. L. "Quad Trees A Data Structure for Retrieval on Composite Keys". Acta Informatica. Springer-Verlag. 4: 1–9, (1974)
33. Zhang J, You S, Gruenwald L. Parallel QuadTree coding of large-scale raster geospatial data on gpgpus. En Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, New York, pp 457–460, (2011)
34. Bentley J. L. Multidimensional binary search trees used for associative searching. Commun. ACM 18(9):509–517, (1975)
35. Castro J, Burns S. Online Data Visualization of Multidimensional Databases Using the Hilbert Space Filling Curve, Lecture Notes in Computer Science, vol 4370, Springer Berlin Heidelberg, chap 9, pp 92–109, (2007)
36. Guttman A. R-trees: a dynamic index structure for spatial searching. SIGMOD Rec 14(2):47–57, (1984)