Métodos de Acceso para Bases de Datos Métrico-Temporales

Anabella De Battista[1], Andrés Pascal[1], Norma Edith Herrera[2], Gilberto Gutiérrez[3]

[1] Dpto. Ing. en Sist. de Información, Universidad Tecnológica Nacional, Entre Ríos, Argentina,
{debattistaa, pascala}@frcu.utn.edu.ar
[2] Dpto. de Informática, Universidad Nacional de San Luis, Argentina,
nherrera@unsl.edu.ar
[3] Universidad del Bío Bío, Facultad de Ciencias Empresariales, Chillán, Chile,
ggutierr@ubiobio.cl

Resumen Las bases de datos métrico-temporales constituyen un nuevo modelo de bases de datos que combina espacios métricos con bases de datos temporales, con el fin de procesar consultas por similitud restringidas a un intervalo o a un instante de tiempo. Uno de los índices propuestos que ha demostrado ser competitivo para resolver este tipo de consultas es el *Historical FHQT*. En este artículo presentamos una mejora a este índice que consiste en permitir el uso de diferentes grupos de pivotes para árboles que corresponden a instantes de tiempo consecutivos. Los resultados experimentales muestran que esta modificación permite mejorar la capacidad de filtrado del índice.
Palabras Claves: Espacios Métricos, Bases de Datos Temporales, Índices, Bases de Datos Métrico-Temporales.

1. Introducción

Las bases de datos tradicionales permiten manipular eficientemente información estructurada en registros, donde cada uno posee campos totalmente comparables. Sobre este modelo se aplica el concepto de búsqueda exacta. El hecho de que las bases de datos actualmente permitan almacenar tipos de datos multimedia (imágenes, audio, video, texto), y el hecho de que este tipo de información no pueda estructurarse, implica que los modelos clásicos no son de utilidad en este ámbito. La problemática de almacenamiento y búsqueda en estos tipos de bases de datos difiere notablemente de las bases de datos clásicas en tres aspectos: primero, los datos no son estructurados, esto significa que es imposible organizarlos en registros y campos, segundo, la búsqueda exacta carece de interés y tercero, resulta de interés mantener todos los estados de la base de datos y no sólo el más reciente a fin de poder consultar el instante o intervalo de tiempo de vigencia de dichos objetos. Como solución a esta problemática surgieron modelos que permiten procesar esta clase de datos.

El modelo de *Espacios Métricos* [4] es un modelo de bases de datos que permite la manipulación de objetos multimedia y la realización de búsquedas por similitud sobre los mismos. Este tipo de búsqueda tiene una amplia gama de aplicaciones, como reconocimiento de imágenes y sonido, compresión de texto, biología computacional,

inteligencia artificial y minería de datos, entre otras. El modelo de *Bases de Datos Temporales* [9] permite almacenar y recuperar datos que dependen del tiempo. Mientras que las bases de datos tradicionales tratan al tiempo como otro tipo de dato más, este modelo incorpora al tiempo como una dimensión. El modelo de *Bases de Datos Métrico-Temporales* [5,6,7] combina características de los dos modelos antes mencionados permitiendo realizar consultas por similitud teniendo en cuenta además el aspecto temporal, esto implica buscar objetos similares a uno dado cuyo intervalo de vigencia se superponga con un intervalo aportado en la consulta. Por ejemplo, si se cuenta con un registro fotográfico de las personas que ingresan a un banco y el período de tiempo que permanecieron en el mismo, sería de interés conocer las personas con rasgos similares a una dada que permanencieron en el banco durante un período de tiempo dado. En este trabajo estamos interesados en métodos de acceso (índices) para este último tipo de bases de datos.

Uno de los índices que permiten procesar eficientemente consultas sobre bases de datos métrico-temporales es el *Historical Fixed Height Queries Tree (H-FHQT)*, que consiste en una lista de instantes válidos donde cada instante contiene el índice métrico *Fixed Height Queries Tree(FHQT)* con todos los objetos vigentes en dicho instante. En este artículo presentamos una mejora a este índice que resultó eficiente principalmente para las consultas métrico-temporales por intervalos.

El artículo está organizado de la siguiente manera. En la Sección 2 se expone una reseña del trabajo relacionado, definiendo los conceptos necesarios para la comprensión de este trabajo. En la Sección 3 presentamos nuestro aporte, el índice *Pivot H-FHQT*. En la Sección 4 se muestra la evaluación experimental de dicho índice y finalmente en la Sección 5 se plantean las conclusiones y el trabajo futuro.

2. Trabajo relacionado

En esta sección daremos una reseña del modelo de espacios métricos, del modelo métrico-temporal y de los índices que forman la base para el desarrollo de este trabajo.

2.1. Espacios Métricos

Los espacios métricos constituyen un modelo de bases de datos que dan el marco teórico necesario para el estudio del almacenamiento y posterior consulta de datos no estructurados. Formalmente un *espacio métrico* se define como un par (U, d) donde U es el universo de objetos válidos del espacio y $d : U \times U \rightarrow R^+$ es una función de distancia definida entre los elementos de U que mide su similitud; esto significa que a menor distancia más cercanos o similares son los objetos. La función d cumple con las propiedades características de una función métrica: $\forall x, y \in U,\ d(x, y) \geq 0$ (positividad), $\forall x, y \in U,\ d(x, y) = d(y, x)$ (simetría) y $\forall x, y, z \in U,\ d(x, y) \leq d(x, z) + d(z, y)$ (desigualdad triangular). Llamaremos base de datos a cualquier subconjunto finito $X \subseteq U$ cuya cardinalidad es $|X| = n$.

Una de las consultas por similitud típica en espacios métricos es la búsqueda por rango, que denotaremos con $(q, r)_d$. Dado un elemento $q \in X$, al que se denomina *query*, y un radio de tolerancia r, una búsqueda por rango consiste en recuperar los objetos de la base de datos que estén a distancia a lo más r de q.

El tiempo total de resolución de una búsqueda contiene tres términos, a saber: *T= # evaluaciones (d)* $\times$ *complejidad (d) + tiempo extra de CPU + tiempo de I/O* . En muchas aplicaciones la evaluación de la función d es tan costosa que las demás componentes de la fórmula anterior pueden ser despreciadas y, en consecuencia, en estos casos la medida de complejidad es la cantidad de evaluaciones de la función de distancia d; este es el modelo de complejidad que usaremos en este trabajo. Para resolver búsquedas por similitud con mayor eficiencia que *O(n)* evaluaciones de distancias, que sería el costo de recorrer secuencialmente la base de datos, se utilizan estructuras de datos o índices que permiten ahorrar cálculos durante el proceso de búsqueda.

En [4] se presenta un desarrollo unificador de las soluciones existentes en la temática. En dicho trabajo se muestra que todos los enfoques para la construcción de índices en espacios métricos consisten en particionar el espacio en clases de equivalencia e indexar dichas clases. Luego, durante la búsqueda descartar algunas clases por medio del índice y buscar exhaustivamente en las restantes. La diferencia entre los distintos algoritmos radica en cómo construyen esta relación de equivalencia. Se pueden distinguir dos grupos: *algoritmos basados en pivotes* y *algoritmos basados en particiones compactas*.

El *Fixed Height Queries Tree*(FHQT), presentado en [1], pertenece al grupo de algoritmos basados en pivotes y básicamente es una variante del *Fixed Queries Tree (FQT)* [2] en la que todas las hojas se encuentran a la misma altura. Originalmente estas estructuras fueron propuestas para funciones de distancias discretas, pero se pueden adaptar a distancias continuas discretizando los valores de las mismas [8].

En un FHQT el árbol se construye a partir de un elemento p (pivote) que puede ser elegido arbitrariamente, o mediante algún procedimiento de selección de pivotes del universo U [3]. Para cada distancia i se crea el conjunto C_i formado por todos aquellos elementos de la base de datos que están a distancia i de p. Luego, para cada C_i no vacío se crea un hijo del nodo correspondiente a p, con rótulo i, y se construye recursivamente un FHQT teniendo en cuenta que todos los subárboles del mismo nivel usarán el mismo pivote como raíz. Este proceso recursivo se continúa hasta lograr que todas las hojas tengan menos de b elementos y estén en un mismo nivel. Notar que la cantidad de pivotes total a utilizar queda determinada por la altura del árbol. La Figura 1 muestra un ejemplo de un FHQT con dos pivotes. Sobre la izquierda se muestra la división del espacio que produce la elección de u_{11} como pivote y sobre la derecha el FHQT que resulta de elegir a u_{11} y u_5 como pivotes. También se ilustra una query q.

Llamaremos *firma* de la query q al vector $(d(q, p_1), d(x, p_2), \ldots, d(q, p_k))$ donde k es la cantidad de pivotes utilizados. Ante un consulta $(q, r)_d$ se utiliza la desigualdad triangular y la firma de q para descartar elementos del espacio sin medir su distancia real a q. Se comienza por la raíz y se descartan todas aquellas ramas con rótulo i tal que $i \notin [d(p, q) - r, d(p, q) + r]$ siendo p el pivote utilizado en la raíz. La búsqueda continúa recursivamente en todos aquellos subárboles no descartados, utilizando el mismo criterio. Los elementos que no pueden ser descartados por este proceso forman parte de una lista de candidatos que posteriormente se deben comparar con la query q para decidir si forman o no parte de la respuesta. Es decir, la lista de candidatos contiene la respuesta real a la consulta más falsos positivos que los pivotes no lograron descartar. Notar que para resolver una consulta se realizarán $k + m$ evaluaciones de distancia donde m es la cardinalidad de la lista de candidatos.

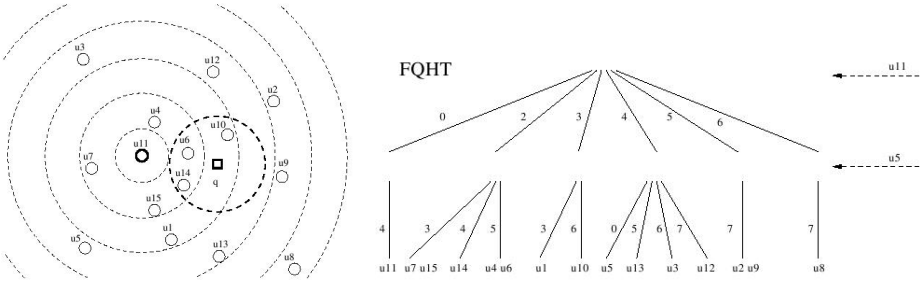

Figura 1. Un ejemplo de un FHQT construido sobre un conjunto de 15 puntos, discretizando los valores de la función de distancia.

2.2. Bases de Datos Métrico-Temporales

Este tipo de bases de datos permite realizar búsquedas sobre objetos no estructurados que tienen un intervalo de vigencia asociado, por lo cual tiene sentido realizar consultas por similitud donde se considere también el aspecto temporal.

Formalmente un *Espacio Métrico-Temporal* se define formalmente como un par (U, d), donde $U = O \times N \times N$ es el universo de objetos y la función de distancia d es de la forma $d : O \times O \rightarrow R^+$. Cada elemento $u \in U$ es una triupla (obj, t_i, t_f), donde obj es un objeto (por ejemplo, una imagen, sonido, cadena, etc.) y $[t_i, t_f]$ es el intervalo de vigencia de obj. La función de distancia d, que mide la similitud entre dos objetos, cumple con las propiedades de una métrica (positividad, simetría, reflexividad y desigualdad triangular).

Una *consulta métrico-temporal* se define como una 4-upla $(q, r, t_{iq}, t_{fq})_d$, tal que: $(q, r, t_{iq}, t_{fq})_d = \{o/(o, t_{io}, t_{fo}) \in X \wedge d(q, o) \leq r \wedge (t_{io} \leq t_{fq}) \wedge (t_{iq} \leq t_{fo})\}$.

Una forma trivial de resolver una consulta métrico-temporal, sin realizar una búsqueda exhaustiva en la bases de datos, es construir un índice métrico agregándole a cada objeto un intervalo temporal que represente la vigencia del mismo. Luego, ante una consulta $(q, r, t_{iq}, t_{fq})_d$ en primer lugar se utilizará el índice métrico para descartar los objetos obj que están a distancia mayor que r de q; y posteriormente se realizará un recorrido del conjunto de elementos no descartados en el primer paso para determinar qué objetos conforman la respuesta a la consulta, que serán aquellos cuyo intervalo de vigencia se superpone con $[t_{iq}, t_{fq}]$. La principal desventaja de esta solución trivial es que no se utiliza la componente temporal para mejorar la capacidad de filtrado inicial del índice, en este proceso sólo se aprovecha la componente métrica. Una mejor estrategia sería que durante el proceso de búsqueda se utilice tanto la componente métrica como la componente temporal para descartar elementos.

El *Historical-FHQT* (H-FHQT) [6] es un índice métrico-temporal que utiliza tanto la componente métrica como la temporal para resolver eficientemente búsquedas métrico-temporales. Este índice consiste en una lista de instantes válidos donde cada uno contiene un FHQT que indexa a todos los objetos vigentes en dicho instante. Los FHQT tienen distintas profundidades, es decir distintas cantidades de pivotes, en función de la cantidad de elementos que deban indexar. La cantidad de pivotes utilizada en

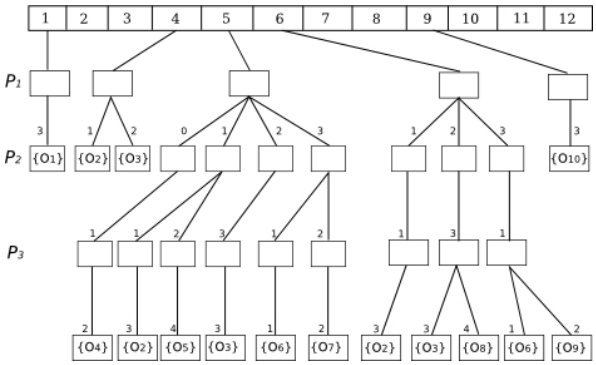

Figura 2. Un ejemplo de un H-FHQT.

cada árbol se calcula como $\lceil log_2(|o_i|) \rceil$, donde $|o_i|$ es la cantidad de objetos vigentes en el instante i. De esta manera se evita que haya árboles con mayor profundidad de la necesaria, con el fin de que la estructura no tenga un costo excesivo en almacenamiento. Si bien en un H-FHQT la cantidad de pivotes en distintos instantes de tiempo varía, siempre se trabaja sobre el mismo conjunto base de pivotes; esto significa que si en el instante i necesito k_i pivotes y en el instante j necesito k_j pivotes, con $k_i < k_j$, entonces los primeros k_i pivotes de ambos instantes son iguales. Con esto se evita que una query deba compararse con grupos distintos de pivotes en distintos instantes, lo que implicaría mayor cantidad de evaluaciones de la función de distancia d para poder calcular la firma de q en cada instante.

La Figura 2 muestra un ejemplo de un H-FHQT construido sobre el intervalo de tiempo [1..12]. Se puede observar que el objeto o_2 está vigente en el intervalo de tiempo [4..6] y se encuentra a distancia 1 de p_1, a distancia 1 de p_2 y a distancia 3 de p_3.

Vamos a denotar con *fhqt*$_i$ al FHQT del instante i y vamos a denotar con k_i a la cantidad de pivotes de *fhqt*$_i$. Una consulta métrico-temporal $(q, r, t_{iq}, t_{fq})_d$ se resuelve en el H-FHQT de la siguiente manera: se seleccionan los instantes i incluidos en el intervalo de consulta, se realizan consultas por rango sobre cada uno de los *fhqt*$_i$ involucrados y se unen los conjuntos resultantes. La Figura 3 muestra el pseudocódigo del algoritmo de consulta. El proceso *extender* es el encargado de ir calculando la firma de la query según se vaya necesitando. Para ello se fija si k_i es mayor que la máxima cantidad de pivotes utilizados hasta el momento (max) en cuyo caso extiende la firma a k_i pivotes; caso contrario la firma ya contiene toda la información necesaria para que el proceso *Rango* realice una búsqueda por rango sobre el *fhqt*$_i$.

3. Mejorando el desempeño del H-FHQT: Pivot H-FHQT

Tal como lo mencionáramos en la sección anterior, el objetivo del usar el mismo grupo base de pivotes para los distintos *fhqt* involucrados en un H-FHQT es evitar el cálculo de distintas firmas en los distintos instantes de tiempo involucrados. Supongamos que un objeto o que no pertenece al resultado de una consulta $(q, r, t_{iq}, t_{fq})_d$, está vigente en varios instantes de tiempo i que pertenecen al intervalo de la consulta. Si el objeto o no pudo ser descartardo por el *fhqt*$_i$, entonces la única posibilidad

```
H-FHQT (q, r, t_iq, t_fq, d) : set
 1. Result= ∅
 2. last = 0
 3. for t_iq ≤ i ≤ t_fq
 4.     if k_i > last
 5.         firma_q = extender(firma_q, last, k_i)
 6.         last = k_i
 7.     end if
 8.     Result= Result ∪ Rango(q, r, d, fhqt_i, firma_q, k_i)
 9. end for
 10. return(Result)
Observación 1: fhqt_i es el FHQT del instante i y k_i es la cantidad
       de pivotes de fhqt_i
Observación 2: Rango realiza la búsqueda (q, r)_d sobre fhqt_i usando las
       primeras k_i componentes del vector firma_q.
```

Figura 3. Pseudocódigo del algoritmo de consulta del H-FHQT

de que sea descartado por el *fhqt*$_{i+1}$ es que k_{i+1} sea mayor que k_i y que esos pivotes adicionales del *fhqt*$_{i+1}$ logren eliminar a o (recordar la regla de eliminación del algoritmo de búsqueda visto en la sección 2.1). Esto significa que la capacidad de filtrado del *fhqt*$_{i+1}$ frente a ese objeto o se reduce a la capacidad de filtrado de los pivotes adicionales, si es que éstos existen.

Lo anterior nos indica que usar conjuntos disjuntos de pivotes para *fhqt* consecutivos, si bien aumenta la cantidad de evaluaciones de distancias al momento de calcular la firma de la query q, también aumenta la probabilidad de disminuir la cantidad de falsos positivos en la lista de candidatos con los que deberá compararse q.

Estas ideas fueron la base que permitieron diseñar las modificaciones al H-FHQT y al algoritmo de búsqueda sobre el mismo, que lograron (como mostraremos en la próxima sección) mejorar el desempeño del índice. Llamaremos a esta nueva versión *Pivot H-FHQT* y la denotaremos con PH-FHQT.

En el PH-FHQT el *fhqt*$_i$ se construye con pivotes diferentes al *fhqt*$_{i-1}$ y al *fhqt*$_{i+1}$. Para lograr esto, cada *fhqt* va tomando los primeros pivotes disponibles de una lista global de pivotes $(p_0, p_1, \cdots, p_{m-1})$, que se maneja como una lista circular: el *fhqt*$_1$ usa los primeros k_1 pivotes, *fhqt*$_2$ usa los siguientes k_2 pivotes, y así siguiendo hasta llegar al último pivote disponible de la lista, en cuyo caso se vuelve al inicio de la misma . De esta forma, el pivote j-ésimo del *fhqt*$_i$ es el pivote que se encuentra en la posición $((k_1 + k_2 + \cdots + k_{i-1} + j) \text{ MOD } m)$ de la lista. La construcción de los *fhqt* consecutivos con diferentes grupos de pivotes da a la estructura mayor poder de filtrado de elementos desde el punto de vista métrico.

En la Figura 4 se presenta un ejemplo de un PH-FHQT construido sobre el intervalo de tiempo $[1, 12]$. Sólo hay objetos vigentes en los instantes 5, 7, 8 y 9. Se puede apreciar que los *fhqt* tienen diferentes profundidades dependiendo de la cantidad de objetos que indexan.

Para realizar una consulta sobre el PH-FHQT podríamos proceder de manera similar a un H-FHQT, esto es: seleccionar los instantes i incluidos en el intervalo de consulta, realizar consultas por rango sobre cada uno de los *fhqt*$_i$ involucrados y unir los conjuntos resultantes. Pero ahora, como la capacidad de filtrado del *fhqt*$_i$ es independiente de la capacidad de filtrado del *fhqt*$_{i-1}$, podemos refinar aún mas la lista de candidatos. Para

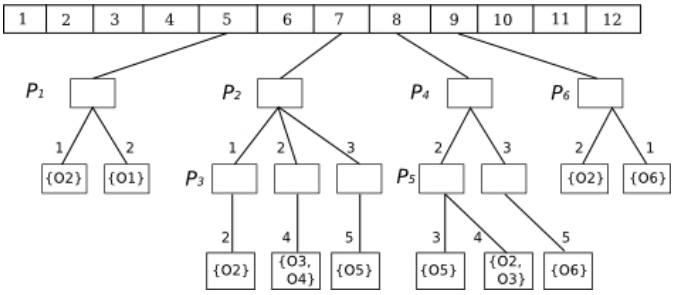

Figura 4. Un ejemplo de un Pivot H-FHQT.

ello, con cada búsqueda en el *fhqt*$_i$ obtendremos una lista C_i de elementos candidatos y una lista D_i formada por los elementos que el *fhqt*$_i$ logró descartar. Para que un objeto o sea un candidato real a formar parte de la respuesta debe pertenecer al menos a alguno de los C_i involucrados y a ninguno de los D_i, es decir, el objeto o debe haber sobrevivido a todos los pivotes de los *fhqt*$_i$ involucrados.

La Figura 5 muestra el pseudocódigo del algoritmo de consulta. La variable $firma_q$ es un vector de exactamente m componentes en el que se van almacenado las firmas ya obtenidas; $first$ y $last$ mantienen la primera y la última posición de $firma_q$ que ya han sido calculadas. El proceso *actualizar* es el encargado de analizar si la firma de la query para el *fhqt*$_i$ ya está en el vector $firma_q$ o debe calcularse y almacenarse en el mismo. Este mecanismo evita recalcular distancias a pivotes que ya hayan sido evaluadas cuando se consultó por similitud en algunos de los FHQT de instantes anteriores. El proceso *Buscar* no hace una búsqueda por rango completa sobre el *fhqt*$_i$, sólo obtiene los candidatos C_i y los descartados D_i. Al finalizar las búsquedas sobre todos los *fhqt*$_i$ involucrados se decide cuáles son los candidatos reales a ser comparados con q.

4. Resultados Experimentales

Para la evaluación experimental se utilizaron dos bases de datos de imágenes vectorizadas ampliamente empleadas por la comunidad de espacios métricos: *COLORS*,

PH-FHQT $(q, r, t_{iq}, t_{fq}, d)$ **: set**
1. $C = D = Result = \emptyset$
2. $first = last = 0$
3. for $t_{iq} \leq i \leq t_{fq}$
4. $\quad firma_q =$ actualizar$(firma_q, first, last, k_i)$
5. $\quad (C_i, D_i) =$ Buscar$(q, r, d, fhqt_i, firma_q, i, k_i)$
6. $\quad C = C \cup C_i$
7. $\quad D = D \cup D_i$
8. end for
9. $C = C - D$
10. for all $o \in C$
11. $\quad$ if $d(q, o) \leq r$ then Result = Result $\cup\{o\}$
12. end for
13. return(Result)

Figura 5. Pseudocódigo del algoritmo de consulta del PH-FHQT.

que contiene vectores de 761 componentes, y *NASA*, que contiene vectores de 20 componentes, disponibles en el sitio *http://www.sisap.org/library/dbs/vectors/*. A partir de ellas se generaron aleatoriamente lotes de tamaños 5.000, 10.000 y 15.000 tanto para *COLORS* como para *NASA*.

A cada objeto se le asignó un identificador y un intervalo de vigencia que indica el período de validez del objeto, siendo [1.. 1000] el intervalo total considerado. Para cada base de datos se utilizó como función de distancia la distancia euclidiana. De aquí en adelante se hará referencia a las bases de datos métrico-temporales generadas como *ColorsMT* y *NasaMT*.

Para cada una de las 6 bases de datos creadas, se generaron 100 consultas por rango métrico-temporales que fueron obtenidas tomando aleatoriamente 100 elementos del lote considerado, variando los radios e intervalos de consulta. Para los radios se utilizaron los valores $r = 5, 9, 11$ para *ColorsMT* y $r = 7, 9, 11$ para *NasaMT*; estos radios de búsquedas retornan aproximadamente el 1 %, 5 % y 10 % de objetos de la base y fueron establecidos experimentalmente. Para los intervalos de consulta se utilizaron los valores: instantánea, 10 %, 25 % y 50 % del total.

Para el H-FHQT, los $fhqt_i$ correspondientes a cada instante de tiempo se construyeron tomando pivotes de una lista construida con elementos de la base de datos elegidos al azar. Para el caso del PH-FHQT se generó la lista global de pivotes con una cardinalidad que varía entre $\lceil log_2(|O_i|) \rceil$ y $\lceil log_2(|O_i|) \rceil * 2$.

Todas las figuras que se presentan a continuación fueron obtenidas promediando los resultados obtenidos con cada una de las 100 consultas realizadas. Por cuestiones de espacio sólo se muestran las gráficas que se consideran más representativas.

La Figura 6 muestra los resultados obtenidos con la base de datos *ColorsMT* de 5.000 objetos con los índices H-FHQT y PH-FHQT para consultas instantáneas (arriba izquierda) y por intervalos. En el eje X de las gráficas se representan los radios de búsqueda y en el eje Y el número promedio de evaluaciones de distancia. Como puede observarse, el índice PH-FHQT tiene un mejor desempeño que el H-FHQT logrando reducciones en la cantidad de evaluaciones de distancias de hasta el 12 % para la consulta instantánea. Esta mejora se acentúa más en el caso de las consultas por intervalos llegando a realizar hasta un 26 % menos de evaluaciones de distancia. En todos los casos los porcentajes de mejora disminuyen a medida que aumentamos el radio de búsqueda llegando a un 7 % en el caso de la consulta instantánea y a un 10 % en las consultas por intervalo. Lo mismo sucedió con los lotes de 10.000 y 15.000 elementos.

La Figura 7 muestra los resultados obtenidos con la base de datos NasaMT de 5.000 objetos. En este caso se puede apreciar que para consultas instantáneas el PH-FHQT sólo supera al H-FHQT cuando $r = 7$, logrando una mejora del 1 %. Cuando $r = 9$ y $r = 11$ en cambio, el H-FHQT es más competitivo que el PH-FHQT, consiguiendo mejoras del 2 % y 3 % respectivamente. Se observa para estos dos casos que los porcentajes de mejoras se incrementan a medida que aumenta también el radio de búsqueda. En cambio cuando se analizan los resultados para las consultas por intervalo se ve que el índice PH-FHQT es el más eficiente en todos los casos. Las mejoras varían entre el 2 % y el 14 %, decreciendo a medida que se incrementa el radio de búsqueda. Esto se repite para los lotes de 10.000 y 15.000 elementos.

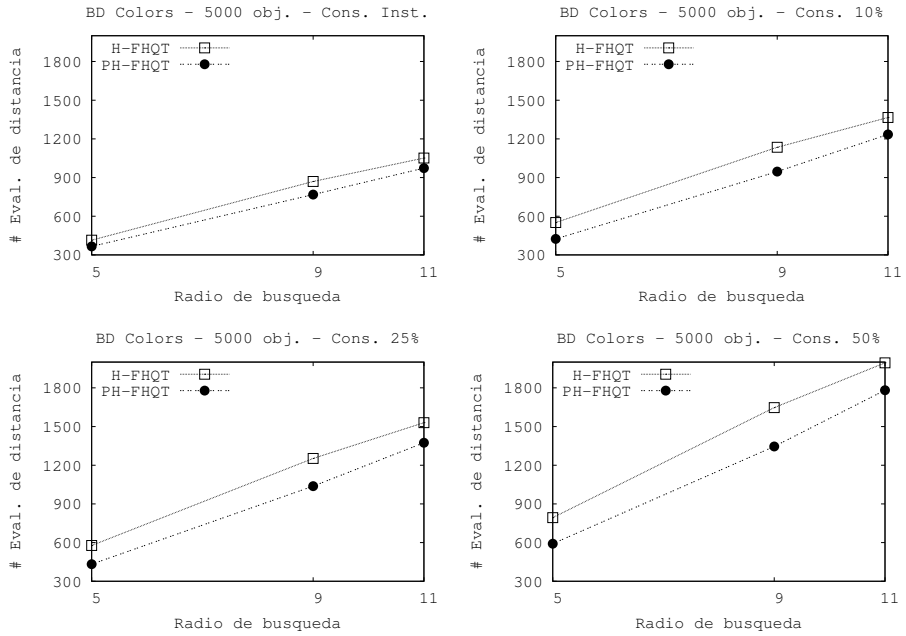

Figura 6. Número medio de evaluaciones de distancia para *ColorsMT* de 5.000 objetos.

5. Conclusiones y Trabajo Futuro

En este artículo presentamos una mejora al índice H-FHQT que consiste en permitir el uso de diferentes grupos de pivotes para árboles que corresponden a instantes de tiempo consecutivos. Esto dió origen a un nuevo índice, el PH-FHQT, que mostró ser más competitivo que su antecesor, el H-FHQT, en la totalidad de las consultas por intervalo ejecutadas sobre las bases *ColorsMT* y *NasaMT*. Para las consultas instantáneas sobre *ColorsMT*, el PH-FHQT superó en todos los casos al H-FHQT y para la base *NasaMT* el índice PH-FHQT fue más competitivo en el 66 % de las pruebas . Las mejoras observadas en este índice se deben al mayor poder de filtrado que se logró generando $fhqt_i$ consecutivos con diferentes grupos de pivotes.

Con respecto al trabajo futuro nos proponemos mejorar este índice respecto del espacio de almacenamiento requerido. El objetivo es detectar si un subárbol del instante i está también en el instante j (con $j > i$), en cuyo caso el instante j debería reutilizar el subárbol del instante i en lugar de crearlo de nuevo. Esto implica diseñar un algoritmo que permita detectar subárboles isomorfos.

Referencias

1. R. Baeza-Yates. *Searching: An algorithmic tour*, volume 37. Allen Kent and James G. Williams, editors, 1997.

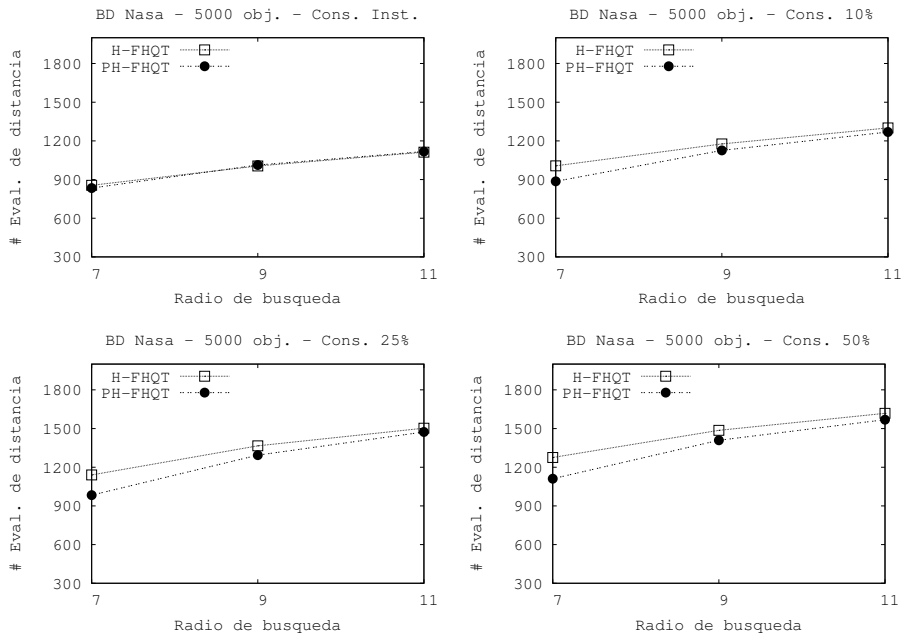

Figura 7. Número medio de evaluaciones de distancia para *NasaMT* de 5.000 objetos.

2. R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *CPM '94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 198–212, London, UK, 1994. Springer-Verlag.
3. B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. In *XXI Conf. of the Chilean Computer Science Society*, pages 33–40, 2001.
4. E. Chávez, G. Navarro, R. Baeza-Yates, and J.L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.
5. A. De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Búsquedas en bases de datos métricas-temporales. In *Actas del VIII Workshop de Investigadores en Ciencias de la Computación*, Buenos Aires, Agentina, 2006.
6. A. De Battista, A. Pascal, G. Gutierrez, and N. Herrera. Un nuevo índice métrico-temporal: el historical-fhqt. In *Actas del XIII Congreso Argentino de Ciencias de la Computación*, Corrientes, Agentina, 2007.
7. A. Pascal, A. De Battista, G. Gutierrez, and N. Herrera. Procesamiento de consultas métrico-temporales. In *XXIII Conferencia Latinoamericana de Informática*, pages 133–144, San José de Costa Rica, 2007.
8. C. Ruano, E. Chávez, and N. Herrera. Discretización binaria del fqtrie. In *Actas del X Congreso Argentino de Ciencias de la Computación*, pages 100–111, Argentina, 2004.
9. B. Salzberg and V. Tsotras. A comparison of access methods for temporal data. *ACM Computing Surveys*, 31(2), 1999.