Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ciencias Físico Matemáticas
Mat. Luis Manuel Rivera Gutiérrez

Selección Efectiva de Permutantes como Elementos Clave en Permutaciones Extendidas para la Búsqueda por Similitud

Por

Victor Ricardo Avila Luna

Tesis para optar al grado de Licenciado en Ciencias Físico-Matemáticas

Asesor	:	**Karina Mariela Figueroa Mora**
Miembros de la comisión	:	Francisco Javier Dominguez Mota
	:	Luis Valero Elizondo
	:	Cuauhtemoc Rivera Loaiza
	:	Jose Carlos Cortes Zavala

Morelia, Michoacán, México.
Julio de 2021

Resumen

El aumento del uso de la tecnología de la información hoy en día nos genera una cantidad cada vez más grande de datos. Gran parte de ellos son de tipo multimedia, es decir, imágenes, videos, audios, etc. Lo cual conlleva a la necesidad de poder realizar *búsquedas* en esas enormes cantidades de datos, además, muchas veces el elemento a buscar no se encuentra en la base de datos pero sí puede haber elementos muy similares, así la principal consulta en este tipo de datos son las búsquedas por *similitud*; los usuarios estamos interesados en encontrar elementos similares, por ejemplo, el rostro de una persona, una firma de voz, huellas digitales o detectar alguna especie de animal.

Una forma de encarar el problema es modelarlo como un espacio métrico, donde se utiliza la base de datos y una función de *distancia* que mide la similitud entre los elementos. Una vez modelado se crea una estructura de datos llamada índice que representa una organización de los datos.

En este trabajo se presenta una propuesta para mejorar el desempeño de uno de estos *índices*, el basado en permutaciones. La idea consiste en seleccionar algunos elementos de la base de datos y llamarlos *permutantes*. El resto de los elementos mide su distancia a los permutantes y los ordena de forma ascendente, a este orden se llamará permutación. La hipótesis es que elementos no tan cercanos pero ubicados en cierta posición pueden llegar a tener permutaciones muy similares o la misma, lo que se busca es poder diferenciar dichos elementos agregando una información extra. El estudio que se presenta en esta tesis consiste en agregar a la permutacion información sobre la distancia que existe entre el conjunto de permutantes y así comparando con el elemento se pueda obtener mejor información sobre la distancia hacia la consulta dada[1].

[1] Búsquedas, Permutantes, Distancia, Similitud, Índices.

Abstract

The increase in the use of information technology today generates an increasingly large amount of data. Most of them are multimedia, that is, images, videos, audios, etc. Which leads to the need to be able to search in these huge amounts of data, in addition, many times the element to be searched is not in the database but there may be very similar elements, thus the main query in this type of data they are searches by similarity; users are interested in finding similar elements, for example, a person's face, a voice signature, fingerprints or detecting some kind of animal.

One way to approach the problem is to model it as a metric space, using the database and a distance function that measures the similarity between the elements. Once modeled, a data structure called an index is created that represents an organization of the data.

This paper presents a proposal to improve the performance of one of these indices, the one based on permutations. The idea is to select some elements from the database and call them permutants. The rest of the elements measures their distance from the permutants and orders them in ascending order, this order will be called permutation. The hypothesis is that elements not so close but located in a certain position may have very similar permutations or the same, what is sought is to be able to differentiate said elements by adding extra information. The study presented in this thesis consists of adding to the permutation information about the distance that exists between the set of permutants and thus comparing with the element, better information about the distance to the given query can be obtained

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Actualmente el desarrollo de las tecnologías de la información y la efectividad del uso de ellas genera cantidades de datos enormes. Lo cual implica querer hacer búsquedas de información en éstos. Hacer búsquedas en una base de datos teniendo conocimiento previo sobre algoritmos de búsqueda no es complicado a menos que la base de datos sea muy grande, los métodos convencionales suelen no poder llegar al resultado ya sea por falta de memoria o tiempos de ejecución demasiado largos.

En una base de datos convencional que pueda cumplir el modelo relacional, es decir que la información pueda ser separada en campos específicos, por ejemplo, una tabla que guarda nombre, matrícula y dirección, si realizamos la consulta de una matrícula, en la base de datos se obtendrá el nombre de la persona y su dirección pues la búsqueda debe ser exacta.

Sin embargo, existe información que no se puede separar en campos específicos, por ejemplo, imágenes, audios, videos, etc. Las bases de datos que guardan este tipo de información se conocen como Bases de Datos Multimedia (BDM). En este tipo de base de datos es común cuestionamientos como por ejemplo, ¿es posible reconocer una persona aunque haya cambiado su atuendo?, o ¿se puede localizar un animal en específico dentro de un grupo de su misma especie?. Nótese que en las consultas de estas bases de datos puede que no se encuentre exactamente el elemento

de la consulta, sin embargo, es interesante y útil recuperar cierto grupo de elementos muy similares al elemento de la consulta.

La forma más trivial de resolver una búsqueda en una base de datos es de manera secuencial, esto es, comparar uno a uno todos los elementos de la base de datos contra la consulta, lo cual no es muy eficiente para casos donde la cantidad de información es muy grande. Una de las estrategias para evitar revisar toda la base de datos es establecer orden en los datos. De manera que al momento de realizar una consulta no revise todos los datos, a esta estructura se le conoce como índice. Los índices nos permiten avanzar rápidamente entre los elementos de la base de datos.

1.1. Búsqueda por Similitud

La búsqueda por similitud consiste en recuperar los objetos más similares a una consulta dada en una base de datos. Este tipo de búsqueda es el núcleo de muchas aplicaciones de la inteligencia artificial. En estas bases de datos, el mayor problema es resolver consultas donde no es posible aplicar una búsqueda exacta, sin embargo, hay formas definidas por expertos de estimar qué tan similares son los objetos. Además, existen bases de datos que tienen una enorme cantidad de elementos, haciendo impensable resolver búsquedas por similitud a través de un escaneo secuencial. Por lo tanto, una solución es preprocesar la base de datos construyendo un índice que permita responder consultas eficientemente.

El problema de la búsqueda de similitud puede ser abordado usando un espacio métrico, este consta de un espacio (base de datos) y una medida de semejanza entre los elementos de ese espacio. No hay una definición general de esta medida dado que está estrechamente ligada a la aplicación y a las características que se quieran evaluar.

Desafortunadamente los métodos actuales para la búsqueda por simi-

litud sufren de lo que es llamada la maldición de la dimensión [CNBYM01]. Donde todo método de búsqueda por similitud, no importa lo eficiente que sea en dimensiones bajas, termina escaneando el conjunto completo de objetos en dimensiones altas. Las técnicas de reducción de dimensiones son efectivas, pero poseen un extra por encima del sistema cuando los datos son intrínsecamente de dimensiones altas, y la precisión de la clasificación caerá si las distancias en los espacios de dimensiones bajas no son bien preservados.

1.1.1. Aplicaciones

Existen muchas aplicaciones que hacen uso de la búsqueda por similitud, algunos ejemplos son: biología computacional, reconocimiento de patrones, recuperación de información y más. Se describen algunas de estas aplicaciones a continuación.

El proceso de reconocimiento de patrones tiene tres estados principales: segmentación, extracción de características, y clasificación. La segmentación consiste en la extracción de objetos individuales de los datos digitalizados. La extracción de las características consiste en hacer un mapeo del objeto digital sobre un espacio vectorial (usualmente de dimensión alta), donde cada coordenada representa el grado de presencia de cierta característica en el objeto. La clasificación consiste en asignar cada objeto a uno fuera del conjunto predefinido por las clases de objetos.

En la biología computacional se utilizan secuencias (una secuencia de ADN, secuencia de nucleótidos o secuencia genética es una sucesión de letras representando una estructura primaria de una molécula real o hipotética de ADN o banda, con la capacidad de transportar información) que definen objetos, por ejemplo proteínas, ADN, etc. Si las secuencias son modeladas como texto, el problema es llevado a búsquedas de cadenas de texto similares. Dependiendo de lo que se quiera evaluar al comparar secuencias, se utilizan distintas medidas de similitud entre cadenas, por ejemplo, una donde se pueda considerar probabilidades de mutaciones.

Otras aplicaciones son los buscadores de internet, búsquedas de imágenes similares, reconocimiento facial, reconocimiento de huellas digitales; éstas son aplicaciones específicas que no se tratarán en este trabajo, sin embargo es relevante mencionarlas por la importancia que representan en la actualidad.

1.2. Conceptos Básicos

1.2.1. Espacios Métricos

Formalmente, sea $\mathbb{X}$ el universo de objetos válidos y d una función de distancia, el par $(\mathbb{X}, d)$ es llamado espacio métrico, donde $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$ que denotará la medida de semejanza entre los elementos de $\mathbb{X}$. Cuanto más pequeño es el valor de d, más parecidos entre sí son los elementos.

La función de distancia d debe satisfacer las siguientes propiedades; Sean $x, y, z \in \mathbb{X}$.

1. Positividad estricta: $d(x,y) > 0 \ \forall x, y$ con $x \neq y$
2. Reflexividad: $d(x,y) = 0$ si y sólo si $x = y$
3. Simetría: $d(x,y) = d(y,x)$
4. Desigualdad triangular: $d(x,z) \leq d(x,y) + d(y,z)$

La última propiedad es la que permite descartar elementos sin ser comparados directamente contra la consulta siempre y cuando se tengan disponibles las distancias de los elementos. Si la distancia no satisface la propiedad de positividad estricta, entonces el espacio es llamado pseudo-métrico. En un espacio pseudo-métrico

pueden existir objetos diferentes que estén a distancia cero entre sí.

Usaremos un conjunto finito de objetos $\mathbb{U} \subseteq \mathbb{X}$ de tamaño $n = |\mathbb{U}|$. Al conjunto de elementos de interés $\mathbb{U}$ lo llamaremos base de datos, diccionario o simplemente conjunto de elementos.

1.2.2. Tipos de Consultas

Existen básicamente dos tipos de consultas por similaridad. Sea $q \in \mathbb{X}$ el elemento de consulta:

Consulta de rango $R(q,r)$. Consiste en recuperar todos los elementos en un radio r dado a la consulta q, es decir, $R(q,r) = \{u \in \mathbb{U} \mid d(q,u) \leq r\}$.

Consulta de los K vecinos más cercanos $KNN(q)$. Consiste en recuperar los K elementos en $\mathbb{U}$ más cercanos a q. Esto es $A \subseteq \mathbb{U}$ tal que $\forall u \in A,\ \ \forall v \in \mathbb{U} - A,\ \ d(q,u) \leq d(q,v),\ \ K = |A|$.

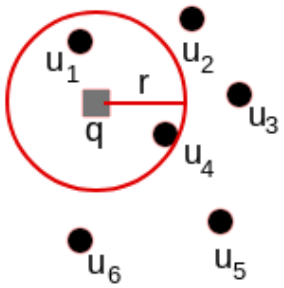

Figura 1.1: Tipo de consulta, consulta por rango

Gráficamente, estas consultas se muestran en las siguientes figuras (un

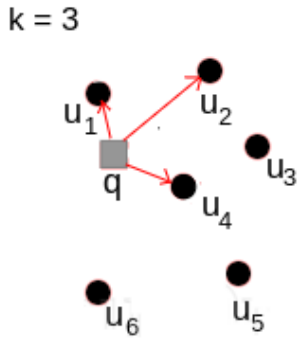

Figura 1.2: Tipo de consulta, consulta k vecinos más cercanos, para k = 3.

espacio métrico en el plano, considerando la distancia Euclidiana), en la figura 1.1 la consulta por rango $R(q,r)$ y en la figura 1.2 la consulta de los k vecinos más cercanos para k = 3, $3NN(q)$.

1.2.3. Tipos de distancias

Los espacios vectoriales m-dimensionales son un caso particular de los espacios métricos, en donde los elementos son identificados con m coordenadas de valores reales. Sean u y q dos vectores con dimensión m, es decir, $u = (u_1, u_2, ..., u_m)$ y $q = (q_1, q_2, ..., q_m)$. La función distancia comúnmente usada es la función L_p definida como:

$$L_p = d_p(u,q) = \sqrt[p]{\sum_{i=1} |u_i - q_i|^p}$$

Donde algunos casos particulares son:

$$\text{Manhattan } L_1 = d_1(u,q) = \sum_{i=1}^{m} |u_i - q_i|$$

$$\text{Euclidiana } L_2 = d_2(u,q) = \sqrt{\sum_{i=1}^{m} |u_i - q_i|^2}$$

$$\text{Máxima } L_\infty = d_\infty(u,q) = \text{máx}_{i=1,..,m} |u_i - q_i|$$

1.2.4. Índice Métrico

Los algoritmos que resuelven consultas por proximidad en espacios métricos generalmente emplean índices para conocer la respuesta sin comparar toda la base de datos uno por uno como una búsqueda secuencial. Un índice es una estructura de datos que facilita la búsqueda de elementos en una base de datos.

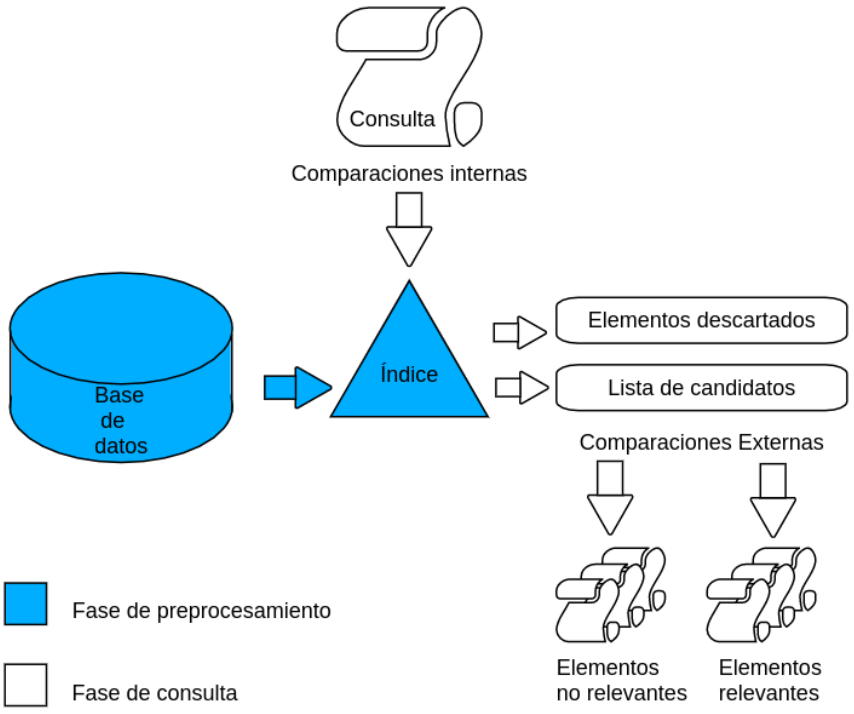

Figura 1.3: Proceso de tratamiento de la base de datos para resolver consultas usando un índice.

Un algoritmo que resuelve consultas por proximidad por lo general tiene dos fases: la de preprocesamiento y la de consulta. En la figura 1.3 se muestra el proceso de preprocesamiento y consulta a una base de datos para obtener el conjunto respuesta. La base de datos se proyecta a un nuevo espacio en el que es posible crear un índice (fase de preprocesamiento). El índice se crea una vez y se almacena. En la parte de la consulta se recorre el índice para conocer la lista de candidatos que serán comparados directamente con la consulta (comparaciones externas) y los elementos que se descartaran de manera segura. Durante el recorrido del índice, es posible que se deban hacer algunas comparaciones por lo que, a éstas las llamaremos comparaciones internas. En la lista de candidatos hay elementos que no son relevantes para la consulta, pero que no pudieron ser distinguidos por el índice

Una proyección de un espacio métrico es una representación reducida del mismo, por ejemplo proyectar el espacio original en un espacio vectorial, mediante una función Φ. De esta manera cada elemento en el espacio original es representado como un punto en el nuevo espacio vectorial. El costo de cada proyección es el número de comparaciones internas. Los dos espacios están relacionados por dos distancias: la original $d(x,y)$ y la distancia del nuevo espacio $D(\Phi(x),\Phi(y))$, donde $x,y \in \mathbb{X}$. Si la proyección $D(\Phi(x),\Phi(y)) \leq d(x,y)$, entonces podemos hacer búsquedas por rango en el espacio proyectado con el radio original. Pero, debido a que pueden ser seleccionados algunos falsos positivos en el espacio proyectado, el resultado obtenido en este espacio es conocido como la lista de candidatos, la que deberá ser verificada en el espacio original para obtener la lista real del conjunto respuesta (comparaciones externas). En un índice, al incrementar las comparaciones internas mejora bastante el poder del filtro y se reducen las comparaciones externas, por lo tanto buscamos un balance óptimo.

Capítulo 2

Trabajo previo

En este capítulo se describen los algoritmos existentes para la búsqueda por similitud en espacios métricos. Las técnicas existentes emplean índices durante las consultas con el propósito de reducir el costo de la búsqueda evitando comparaciones de distancia.

Los índices son estructuras de datos usadas para navegar en los datos y poder tomar decisiones sobre estos, ya sea compararlos o descartarlos. Las propuestas existentes para construir índices consisten en usar ciertos elementos de la base de datos (seleccionados en su mayoría por heurísticas), para calcular y almacenar las distancias entre estos y el resto de la base de datos.

Básicamente el trabajo previo se divide en tres familias de algoritmos: los basados en pivotes, los basados en particiones compactas y los basados en permutaciones. A continuación se analizaran las bases de cada familia.

2.1. Clasificación de algoritmos

2.1.1. Algoritmos basados en pivotes

Para crear un índice basado en pivotes elegimos un subconjunto de elementos llamados 'pivotes' de la base de datos $\mathbb{P} = \{p_1, p_2, ...p_k\} \subseteq \mathbb{U}$, de tamaño $k = |\mathbb{P}|$. Donde para cada elemento de la base de datos se precalculan las distancias hacia los elementos del conjunto $\mathbb{P}$, cuyo conjunto de distancias $\{d(p_1,u), d(p_2,u), ..., (p_k,u)\}$ $\forall u \in \mathbb{U}$, conformará el índice.
Dada una consulta q, se calcula $d(p,q)$ $\forall p_i \in \mathbb{P}$ (comparaciones internas). Con esto se pueden minimizar las distancias entre q y cualquier $u \in \mathbb{U}$ tal como se demuestra en el siguiente lema.

Lema 1. Dados tres objetos $q \in \mathbb{X}$, $u \in \mathbb{U}$, $p \in \mathbb{P}$, sabemos que $|d(q,p) - d(p,u)| \leq d(q,u) \leq d(q,p) + d(p,u)$.
Demostración: El límite superior se obtiene directamente de la desigualdad triangular

$$d(q,u) \leq d(p,q) + d(p,u)$$

En el caso del límite inferior, de acuerdo a la desigualdad triangular, se tiene que:

$$d(p,u) \leq d(p,q) + d(q,u),$$

$$d(p,q) \leq d(p,u) + d(u,q)$$

Estas desigualdades implican:

$$d(p,u) - d(p,q) \leq d(q,u),$$

$$d(p,q) - d(p,u) \leq d(u,q)$$

Combinando estas desigualdades y usando la simetría, obtenemos que:

$$|d(p,u) - d(p,q)| \leq d(u,q) \quad \blacksquare.$$

Los algoritmos que usan este tipo de índices lo hacen de la siguiente manera: durante una consulta por rango, usando el lema 1, se pueden descartar todos los elementos de la base de datos $u \in \mathbb{U}$ tales que $\exists p \in \mathbb{P}$, $|d(q,u) - d(p,u)| > r$. Los elementos no descartados se comparan directamente contra la consulta (comparaciones externas).

Spatial Selection of Sparse (SSS)

El algoritmo Spatial Selection of Sparse (SSS) [RBFP07], una técnica de pivotes que permite lidiar con colecciones dinámicas y funciones continuas de distancia. Es un método dinámico puesto que la colección puede ser inicialmente vacía y crecer o decrecer a medida que más elementos son añadidos o eliminados de la base de datos. La estrategia de selección de pivotes, la cual selecciona dinámicamente un conjunto de pivotes bien distribuidos en el espacio métrico, una propiedad que permite descartar más objetos del resultado cuando resolvemos una búsqueda.

La estrategia de selección de pivote es la siguiente: Sea $(\mathbb{X}, d)$ un espacio métrico, $\mathbb{U} \subseteq \mathbb{X}$ una colección y M la distancia máxima entre cualquier par de objetos, $M = max\{d(x,y)|x,y \in \mathbb{U}\}$. Primero, el conjunto de pivotes se inicia con el primer objeto de la colección, Luego, para cada elemento se elige $x_i \in \mathbb{U}$, x_i como un nuevo pivote si su distancia a cualquier pivote en el actual conjunto de pivotes es igual o más grande que αM, siendo α una constante en que los óptimos valores están alrededor de 0.4. Es decir, un objeto en la colección se vuelve un nuevo pivote si está localizado a más de una fracción de la máxima distancia con respecto a todos los pivotes actuales. Por ejemplo, si $\alpha = 0.5$ un objeto se elige como un nuevo pivote si está localizado a más de una mitad de la distancia máxima de los pivotes actuales.

Los algoritmos basados en pivotes se muestran en [PZB06], [HS00],

[CNBYM01]. Algunos algoritmos interesantes con pivotes se pueden consultar en [BYCMW94], [CMN99].

2.1.2. Algoritmos basados en particiones compactas

Esta Estrategia divide el espacio en zonas tan compactas como sea posible. Selecciona un conjunto de objetos $\mathbb{P} = (p_1, p_2, ..., p_k) \subset \mathbb{U}$, dividiendo el espacio, distribuyendo las k zonas pertenecientes a los p_i, donde los p_i son llamados "centros" de su zona. El índice está compuesto por los centros. Un criterio de distribución, entre varios, es asignar cada u a la zona cuyo p_i sea el más cercano.
En este tipo de algoritmos se dividen de acuerdo a su procedimiento de búsqueda: los que usan radio de cobertura r_c, los que usan hiperplanos y los que usan ambos. El radio de cobertura r_c es la distancia máxima del centro de la zona a los elementos en ella, mientras que un hiperplano es la frontera entre zonas cuando cada elemento es asignado a su centro más cercano.

Lema 2. Dados tres objetos $u, c \in \mathbb{U}, q \in \mathbb{X}$ si c es el centro de la zona a la que pertenece u, con radio de cobertura r_c, sabemos que, $d(q,u)$ está acotada: máx $\{d(c,r) - r_c, 0\} \leq d(q,u) \leq d(p,q) + r_c$.
Demostración. Ambos límites los obtenemos de la desigualdad triangular y de la cota superior:

$$d(p,q) \leq d(p,u) + d(u,q),\ d(p,u) \leq r_c$$

Usando estas dos ecuaciones tenemos:

$$d(p,q) - d(q,u) \leq d(p,u) \leq r_c$$

Esto implica que:

$$d(p,q) - r_c \leq d(q,u)$$

Sabemos que

$$d(q,u) \geq 0$$

Por lo tanto el límite inferior es

$$\text{máx}\{d(q,p) - r_c, 0\} \leq d(q,u) \quad \blacksquare$$

El límite superior lo obtenemos de la desigualdad triangular

$$d(q,u) \leq d(q,p) + d(p,u) \leq d(q,p) + r_c$$

Lema 3. Sea $u \in \mathbb{U}$ el objeto más cercano a p_1 que a p_2 o que equidista, es decir, $d(p_1,u) \leq d(p_2,u)$. Dados $d(q,p_1)$ y $d(q,p_2)$, podemos establecer la cota $\text{máx}\{\frac{d(q,p_1)-d(q,p_2)}{2}, 0\} \leq d(q,u)$.

Demostración. De la desigualdad triangular sabemos que

$$d(q,p_1) \leq d(q,u) + d(p_1,u)$$

Lo cual lleva a

$$d(q,p_1) - d(q,u) \leq d(p_1,u)$$

Del mismo modo tenemos que

$$d(p_2,u) \leq d(q,p_2) + d(q,u)$$

Y por hipótesis

$$d(p_1,u) \leq d(p_2,u).$$

Combinando en las ecuaciones anteriores obtenemos

$$d(q,p_1) - d(q,u) \leq d(p_1,u) \leq d(p_2,u) \leq d(q,p_2) + d(q,u),$$

$$d(q,p_1) - d(q,p_2) \leq 2d(q,u)$$

Finalmente, sabemos que

$$d(q,u) \geq 0$$

Así la cota inferior es:

$$\text{máx}\{\tfrac{d(q,p_1)-d(q,p_2)}{2}, 0\} \leq d(q,u). \quad \blacksquare$$

En una consulta de rango $(q,r)_d$ esta familia de algoritmos descarta aquellas zonas de centro p tales que $d(p,q) - r_c > r$, pues usando el *lema 2* sabemos que cualquier elemento u en esa zona no cumple con $d(q,u) > r$. En las zonas no descartadas se repite el proceso hasta que se llega a una zona sin divisiones.
Los algoritmos que utilizan esta estrategia se pueden leer en [CN00], [CPZ97], [Ben79], [DN87], [Uhl91], por mencionar algunos.

2.1.3. Algoritmos basado en permutaciones

Esta estrategia presentada en [CFN05] consiste en definir algunos elementos como puntos de interés $\mathbb{P} = \{p_1, p_2, ...p_k\} \subseteq \mathbb{U}$ (llamados *permutantes*). Luego, cada objeto en la base de datos calcula su distancia con todos los permutantes; ordenados de forma creciente, el nuevo orden de los permutantes es precisamente la permutación construida por este elemento. Si dos elemento u y v son iguales sus permutaciones serán iguales, se esperaría que si dos elementos son parecidos, sus permutaciones sean parecidas.

Cuando una consulta q es dada, se compara con $\mathbb{P}$ (son llamadas distancias internas). Al momento de la consulta se calcula Π_q (permutación de la consulta)

para una consulta q usando $\leq_q$ sobre el conjunto $\mathbb{P}$, ordena los Π_u, $u \in U$ de mayor a menor similaridad con respecto a Π_q.

Como medida de similaridad entre permutaciones se usa Spearman Footrule [DG77], se denota por $S_f(\Pi_q, \Pi_u)$, S_f es la suma de las diferencias en las posiciones relativas de cada elemento en las dos permutaciones. Para cada $p_i \in \mathbb{P}$ se calcula su posición en Π_u y Π_q, esto es $\Pi_u^{-1}(p_i)$ y $\Pi_q^{-1}(p_i)$, es decir:

$$S_f(\Pi_q, \Pi_u) = \sum_{i \leq 1 \leq k} |\Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i)|$$

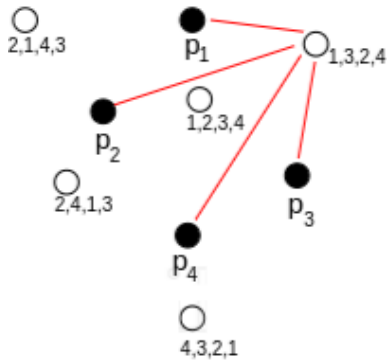

Figura 2.1: Creación del índice, formando la permutación de cada elemento de acuerdo a la distancia contra los permutantes.

En la figura 2.1 y 2.2 se presenta un ejemplo de las 2 fases de este algoritmo. La figura 2.1 muestra la fase de construcción de las permutaciones, y la figura 2.2 muestra la consulta y su permutación. La tabla 2.1 mostrada presenta la comparación entre cada permutación de la base de datos contra la permutación de la consulta.

El siguiente paso es buscar por una lista de candidatos (los elementos más prometedores). Las distancias calculadas entre q y todos los elementos de esta lista se les llama distancias externas. Con suerte, esta lista contendrá sólo unos pocos objetos. En este paso, usualmente, hay una estructura de datos que recupera rápidamen-

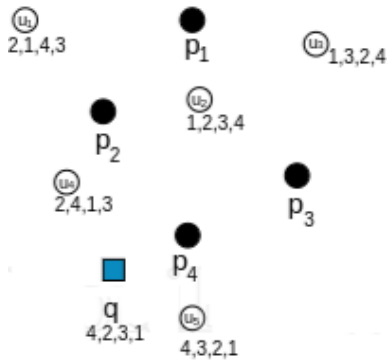

Figura 2.2: Cálculo de permutación de la consulta, Dada la consulta q se calcula su permutación, y se buscan los elementos que tengan la permutación más semejante.

te estos elementos. Nótese que debería haber una compensación entre las distancias internas y externas.

Algunas estructuras de datos propuestas para recuperar la lista de candidatos han sido introducidas, por ejemplo, usando un índice invertido y un árbol de prefijo para encontrar las permutaciones más similares. Ambos algoritmos usan muchos permutantes (512 al menos). Mientras el conjunto de permutantes incrementa, la exhaustividad es mejor, pero la distancia interna también incrementa.

Elemento	Permutación Π_{u_i}	$S_f(\Pi_q, \Pi_{u_i})$
q	4231	
u_5	4321	2
u_1	2143	6
u_2	1234	6
u_4	2413	8
u_3	1324	8

Cuadro 2.1: Tabla de permutaciones ordenadas respecto a la consulta.

Capítulo 3

Propuesta

La técnica de permutación tiene un rendimiento muy competitivo. Es un hecho conocido que, las grandes permutaciones son más efectivas que las cortas, pero, hay un precio que pagar: las distancias internas. Incrementar el tamaño de las permutaciones implica incrementar el número de permutantes y también el número de distancias internas que tienen que ser calculadas.

3.1. Idea principal

Una idea novedosa para extender elementos, v.g. los permutantes, es mezclar la distancia de un elemento hacia los permutantes y la que hay entre ellos mismos. Esto significaría que un elemento no solo guarda el orden de los permutantes, sino también debe guardar la relación de distancia entre los permutantes, a esta idea la llamaremos *permutaciones extendidas*.

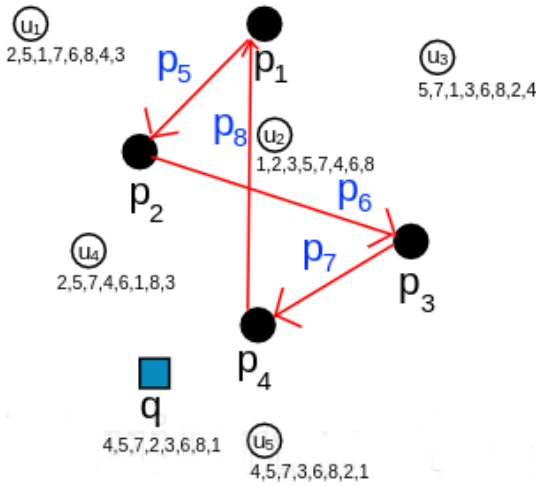

Figura 3.1: Idea de las *permutaciones extendidas*. Las líneas marcadas serán las distancias a agregar $d(p_1, p_2)$, $d(p_2, p_3)$, $d(p_3, p_4)$ y $d(p_4, p_1)$ denotadas como p_5, p_6, p_7 y p_8 respectivamente

3.2. Permutaciones Extendidas

La propuesta presentada en esta tesis consiste en tener una permutación más larga sin tener que calcular distancias extras. Para esto, la propuesta es usar las distancias entre permutantes (al menos una parte de ellas). Es decir, sea $\mathbb{P}$ un conjunto de permutantes, cada elemento calcula sus distancias hacia $\mathbb{P}$. Entonces, para cada elemento $u \in \mathbb{U}$ se obtiene $D(u, \mathbb{P}) = \{d(u, p_1), ..., d(u, p_k)\}$. Sin embargo, previamente, las distancias $d(p_i, p_j)$ fueron calculadas para $1 \leq i \leq k$ y $j = i + 1$; es decir, $d(p_1, p_2), d(p_2, p_3), ..., d(p_{k-1}, p_k)$ y la última distancia podría ser una distancia arbitraria, por ejemplo, $d(p_k, p_1)$. En este caso, tiene solo que tener exactamente el tamaño para cada permutación en $2 \times k$.

En la figura 3.1 se muestra la idea de la propuesta. Hay cuatro puntos como permutantes $\mathbb{P} = \{p_1, p_2, p_3, p_4\}$, y cinco puntos como objetos $\{u_1, u_2, u_3, u_4, u_5\}$. Las líneas rojas representan las distancias entre los permutantes. Es decir, $d_{12} = d(p_1, p_2)$, $d_{23} = d(p_2, p_3)$, $d_{34} = d(p_3, p_4)$ y $d_{41} = d(p_4, p_1)$; estas distancias serán renombradas como p_5, p_6, p_7 y p_8 respectivamente, abusando de la notación. La per-

Elemento	Permutación Π_{u_i}	$S_f(\Pi_q, \Pi_{u_i})$
q	45723681	
u_5	45736821	12
u_4	25746183	14
u_1	25176843	24
u_3	57136824	24
u_2	12357468	28

Cuadro 3.1: Tabla de permutaciones extendidas ordenadas respecto a la consulta.

mutación será $\Pi_{u_1} = (p_2, d_{12}, p_1, d_{34}, d_{23}, d_{41}, p_4, p_3)$ o solo $\Pi_{u_1} = (2,5,1,7,6,8,4,3)$, $\Pi_{u_2} = (1,2,3,5,7,4,6,8)$, $\Pi_{u_3} = (5,7,1,3,6,8,2,4)$, $\Pi_{u_4} = (2,5,7,4,6,1,8,3)$ y finalmente $\Pi_{u_5} = (4,5,7,3,6,8,2,1)$ la tabla 3.1 muestra la permutación de cada objeto y su distancia Spearman Footrule contra la consulta q. Nótese que es la misma consulta y el mismo conjunto de datos de la figura 2.2 a la cual comparando la tabla 2.1 con la 3.1 podemos ver que el objeto u_4 con el algoritmo permutación extendida si muestra una distancia cercana a q.

3.3. Permutaciones Extendidas con Selección SSS

Como se vió en el capítulo 2 un algoritmo presentado en [RBFP07] el cual da una selección de pivotes espacialmente distribuidos en la base de datos, usaremos esa misma estrategia pero ahora para nuestra propuesta de permutantes extendidos.

En la figura 3.2 usando el mismo conjunto de elementos de los ejemplos anteriores seleccionamos los permutantes de acuerdo al algoritmo SSS en este conjunto M es la distancia entre p_1 y p_3 se utilizó $alpha = 0{,}5$. EL algoritmo nos dió tres objetos como permutantes espacialmente distribuidos $\mathbb{P} = \{p_1, p_2, p_3\}$, Luego calculamos sus distancias entre ellos en orden creciente, es decir, $d(p_i, p_{i+1})$ donde $1 \leq i < 3$ y agregamos la distancia arbitraria $d(p_3, p_1)$ estas distancias serán renombradas como p_4, p_5 y p_6. La permutaciones serán $\Pi_{u_1} = (2,1,4,5,3,6)$, $\Pi_{u_2} = (2,1,4,5,3,6)$, $\Pi_{u_3} = (2,4,5,1,3,6)$, $\Pi_{u_4} = (2,1,3,4,5,6)$, $\Pi_{u_5} = (3,2,4,5,1,6)$ y $\Pi_{u_6} = (2,3,4,5,1,6)$. La tabla 3.2 muestra la permutación de cada objeto y su distancia Spearman Footrule

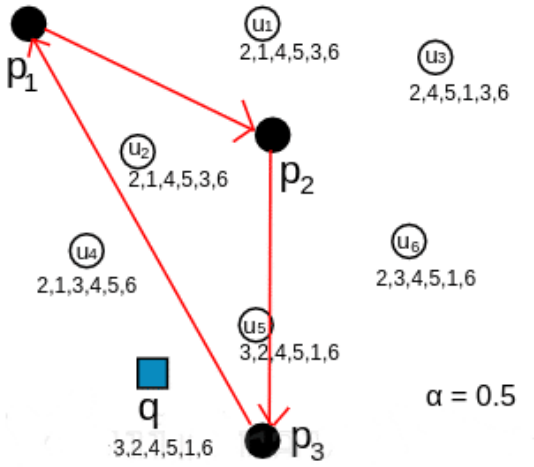

Figura 3.2: Permutaciones Extendidas, usando SSS para la selección de permutantes con $\alpha = 0{,}5$

contra la consulta q. Nótese que el elemento mas cercano, en este caso u_5 tiene la misma permutación que la consulta.

Elemento	Permutación Π_{u_i}	$S_f(\Pi_q, \Pi_{u_i})$
q	324516	
u_5	324516	0
u_6	234516	2
u_2	214536	4
u_1	214536	4
u_4	213456	8
u_3	245136	10

Cuadro 3.2: Tabla de permutaciones extendidas usando SSS ordenadas respecto a la consulta.

Capítulo 4

Experimentación

La búsqueda de similitud consiste en recuperar los objetos más similares de una base de datos cuando se da una consulta. Así que se utilizó una base de datos real, esta consiste en 40,150 vectores, de imágenes de la NASA [DIM]. Cada elemento es un vector de características en $\mathbb{R}^{20}$. La distancia Euclidiana es utilizada para comparar objetos.

La Figura 4.1 muestra el rendimiento en esta base de datos real, el eje *y* muestra el total de distancias internas + externas. Mostramos los resultados considerando $k = 8, 16$ permutantes.

La Figura 4.2 muestra el rendimiento de nuestra propuesta usando $k = 32, 64$ permutantes. En el eje *y* el total de distancias son mostradas internas + distancias externas. La propuesta en este trabajo tiene un mejor rendimiento porque está utilizando menos distancias que otras técnicas.

En orden de comparar nuestra propuesta, probamos los algoritmos: basado en permutantes, permutantes extendidos y permutantes extendidos con SSS. En estas figuras, es posible notar que nuestra propuesta reduce las evaluaciones de distancia en un 59 % y mejora implementando SSS a un 68 % para $NN = 8$, $k = 16$.

En este punto, usar sólo el algoritmo de permutación las distancias internas fueron 16, y 285 distancias externas; Permutación extendida necesita 16 distancias internas 116 distancias externas; finalmente, permutación extendida con SSS usa 16 distancias internas y 88 distancias externas. En este caso, incluso cuando consideramos una permutación con 128 elementos (64 permutantes + las 64 distancias entre permutantes) es un poco mejor porque las distancias internas tienen un costo extra. Por eso, utilizar 64 permutantes no es mejor que basado en permutantes , se observa una mejora al utilizar SSS para la selección de permutantes.

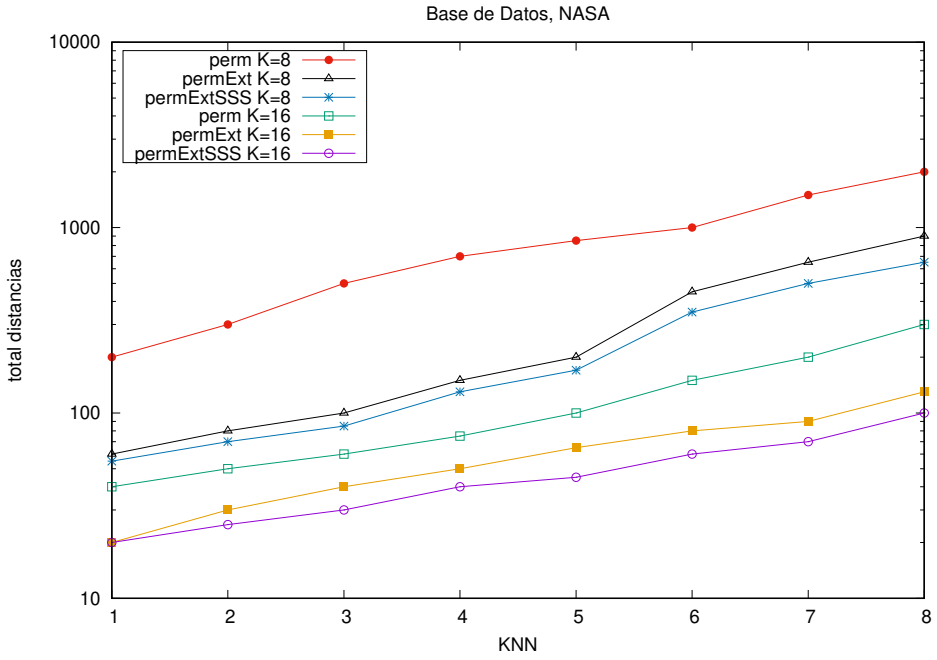

Figura 4.1: BD NASA, gráficas variando los permutantes k=8 y k=16 para recuperar vecinos más cercanos. El eje x corresponde a los resultados de buscar 1 vecino, la segunda fila para 2 vecinos y así sucesivamente, el eje *y* representa la cantidad de distancias realizadas.

La segunda base de datos que probamos consiste de 112,682 histogramas de color, representado como 112-vectores de caractrística dimencional. Fue obtenido del proyecto SISAP para conjuntos benchmark de espacios métricos [SIS]. Escogimos 500 histogramas al azar como el conjunto prueba para consultar y el resto del conjunto de datos como nuestra base de datos para ser indexado.

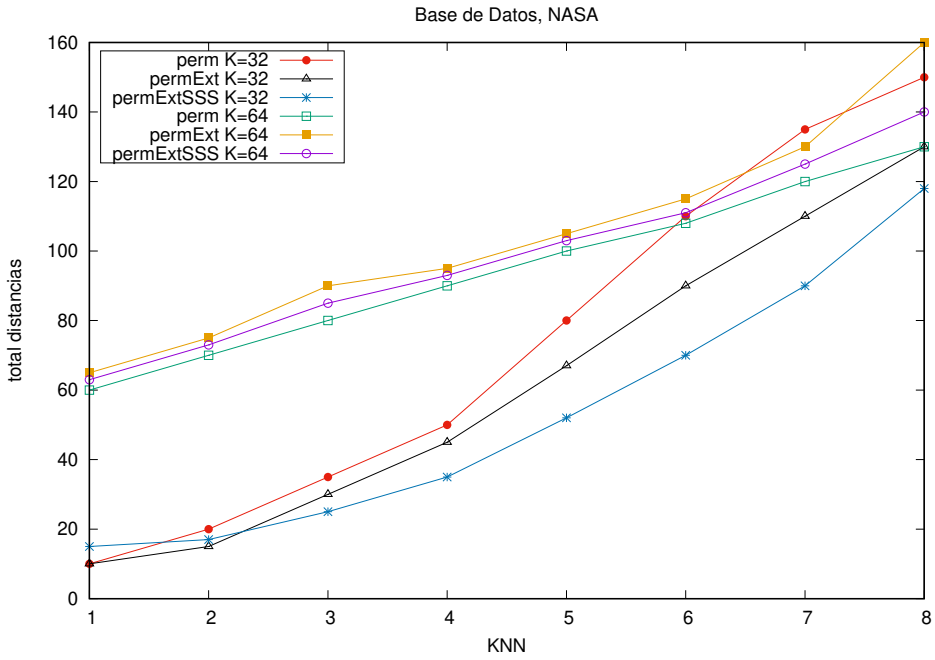

Figura 4.2: BD NASA, gráficas variando los permutantes k=32 y k=64 para recuperar vecinos más cercanos.

La Figura 4.3 muestra el rendimiento de nuestra propuesta usando k = 8, 16 permutantes, y k = 32, 64 en la figura 4.4. En el eje *y* el total de distancias son mostradas (internas + distancias externas). La propuesta en este trabajo tiene un mejor rendimiento porque los permutantes extendidos está utilizando 25 % menos distancias y con SSS hasta 35 % menos distancias que en el algoritmo basado en permutaciones para $NN = 8$, $k = 32$, si usamos $NN = 8$ y $k = 64$, podemos observar que se usa un 41 % menos distancias y 52 % menos con SSS que el algoritmo de permutantes original. En esta base de datos, el rendimiento de nuestra propuesta es excelente cuando k = 64 comparando con la base de datos de la NASA.

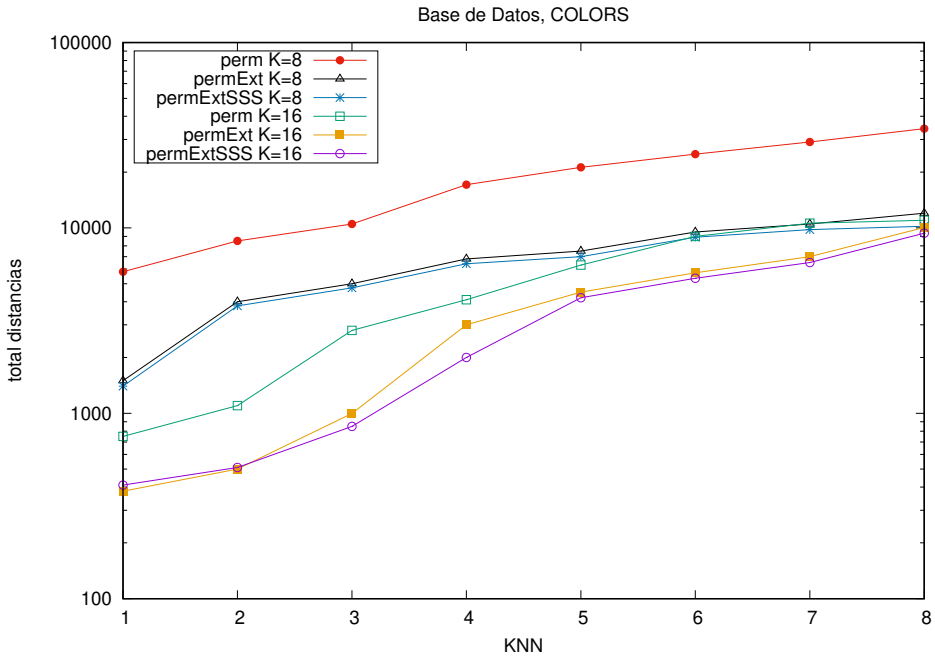

Figura 4.3: BD COLORS, gráficas variando los permutantes k=8 y k=16 para recuperar vecinos más cercanos.

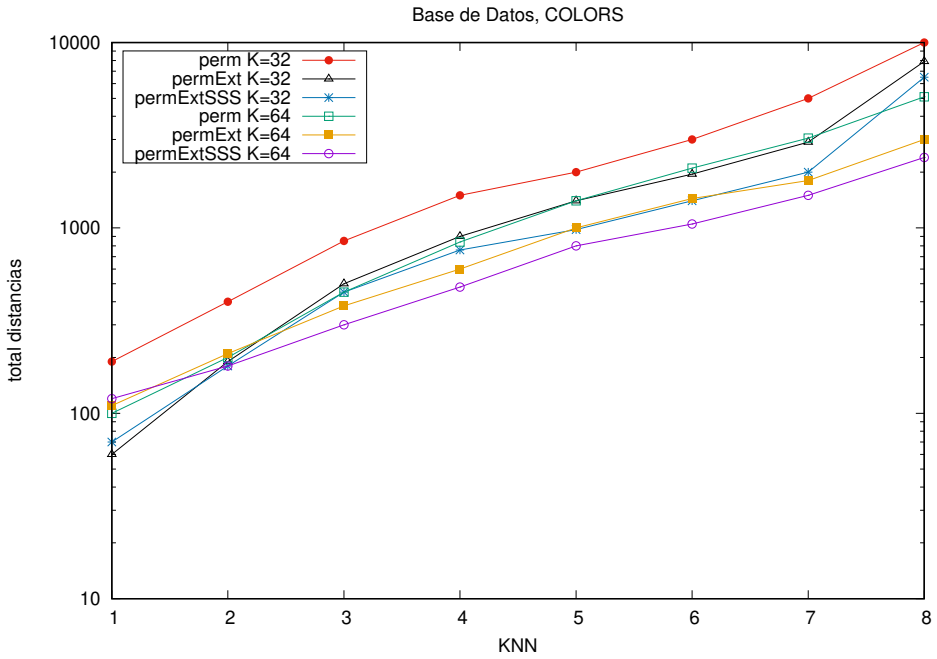

Figura 4.4: BD COLORS, gráficas variando los permutantes k=32 y k=64 para recuperar vecinos más cercanos.

Capítulo 5

Conclusiones

Las búsquedas por similitud reflejan la importancia de obtener los resultados más significativos para consultas de cualquier índole, ya que en la actualidad se está rodeado de la tecnología y se hace uso de ella para diversas tareas. Dada alguna consulta se busca obtener el resultado más relevante para dicha consulta, y este resultado se logra mediante la comparación de la consulta con la base de datos.

En este trabajo se hizo uso de las búsquedas por similaridad en espacios métricos es decir se mapeo el problema a un espacio métrico para así poder hacer uso de la función de distancia lo cual permite la creación de un índice a través del cual se realizan las comparaciones de distancias de la consulta con la base de datos. Se propone agregar información adicional a las permutaciones realizadas en la indexación del algoritmo basado en permutaciones dicha información nos permite poder ubicar espacialmente dos elementos que por su cercanía su permutación es muy parecida y de esa manera poder dar resultados mucho más aproximados a la consulta además se propuso agregar el algoritmo Spatial Selection of Sparse en la selección de los permutantes con muy buenos resultados experimentales.

Los resultados obtenidos en los experimentos con el uso de estas técnicas muestran una mejora en comparación con las técnicas ya conocidas en este caso se puede comparar con el algoritmo basado en permutaciones y se observa la mejora es

menor la cantidad de distancias calculadas.

Como trabajo futuro, estamos interesados en usar otras técnicas de indexación para nuestros permutantes extendidos que permita obtener rendimientos aún más efectivos como lo vimos al utilizar SSS, también otra interesante pregunta es si podemos reducir las distancias extras que agregamos a la permutación buscando las que resulten más eficientes.

Bibliografía

[Ben79] J. Bentley. Multidimensional binary search trees in database applications. *IEEE Transactions on Software Engineering*, 5(4):333–340, 1979.

[BYCMW94] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Proceedings 5th Combinatorial Pattern Matching (CPM'94)*, volume 807 of *Lecture Notes in Computer Science*, pages 198–212, 1994.

[CFN05] E. Chávez, K. Figueroa, and G. Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In *MICAI 2005: Advances in Artificial Intelligence*, volume 3789 of *Lecture Notes in Computer Science*, pages 405–414, 2005.

[CMN99] E. Chávez, J. Marroquín, and G. Navarro. Overcoming the curse of dimensionality. In *European Workshop on Content-Based Multimedia Indexing (CBMI'99)*, pages 57–64, 1999.

[CN00] E. Chávez and G. Navarro. An effective clustering algorithm to index high dimensional metric spaces. In IEEE CS Press, editor, *7th International Symposium on String Processing and Information Retrieval (SPIRE 2000)*, pages 75–86, 2000.

[CNBYM01] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. of the 23rd Conference on Very Large Databases (VLDB'97)*, pages 426–435, 1997.

[DG77] Persi Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):262–268, 1977.

[DIM] DIMACS. http://www.dimacs.rutgers.edu/challenges/sixth/- software.html.

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

[HS00] G. Hjaltason and H. Samet. Incremental similarity search in multimedia databases. *Technical Report TR 4199. Department of Computer Science, University of Maryland*, 2000.

[PZB06] V. Dohnal P. Zezula, G. Amato and M. Batko. *Similarity Search: The Metric Space Approach, volume 32 of Advances in Database System*. Springer, 2006.

[RBFP07] Nora Reyes, Nieves R. Brisaboa, Antonio Farina, and Oscar Pedreira. Spatial selection of sparse pivots for similarity search in metric spaces. 39, 2007.

[SIS] SISAP. http://www.sisap.org/metricspaceslibrary.html.

[Uhl91] J. Uhlmann. Implementing metric trees to satisfy general proximity/similarity queries. Manuscript, 1991.