Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ciencias Físico Matemáticas
"Mat. Luis Manuel Rivera Gutiérrez"

DETECCIÓN DE LA LISTA DE ELEMENTOS CANDIDATOS A PARTIR DE UN ÍNDICE INVERTIDO PARA LA BÚSQUEDA POR SIMILITUD EN ESPACIOS MÉTRICOS

T E S I S

que para obtener el título de:

licenciado en ciencias físico matemáticas

P R E S E N T A :

Elizabeth Ruiz Torres

ASESORA

Dra. Karina Mariela Figueroa Mora

Morelia, Michoacán. Mayo 2021

A mis padres.

Agradecimientos

En este apartado me gustaría expresar mi agradecimiento a cada una de las personas que me ayudaron en este proceso de aprendizaje, tanto académico como personal.

Quiero agradecer de forma especial a mi asesora la Dra. Karina Figueroa Mora por haberme permitido realizar este trabajo de Tesis bajo su dirección. Por su guía, el constante apoyo y su paciencia, además de motivarme para dar siempre lo mejor. Y por contribuir en mi interés por la programación.

También quiero agradecer a mis sinodales, en particular al Dr. Francisco Javier Domínguez Mota y al Dr. Fernando Hernández Hernández, por sus correciones y aportaciones.

Por supuesto, a mis padres Matilde y Jesús, quienes me han apoyado a lo largo de mi vida, haciendo posible lograr mis metas. Por ser grandes ejemplos e inspirarme cada día, por su constante ánimo, comprensión y paciencia, en particular durante este proceso. Así mismo a mi hermano Juan Carlos por su apoyo y consejos.

A mis amigas y amigos, en especial a Francis, Brenda, Liz, Dani y Andrés, quienes hicieron el trayecto más agradable y divertido, les agradezco por brindarme su amistad, por todas las experiencias que pasamos juntos y su apoyo en los momentos difíciles.

Índice general

I INTRODUCCIÓN **1**

1. Introducción **2**

1.1. Aplicaciones 3

II CONCEPTOS BÁSICOS **4**

2. Conceptos básicos **5**

2.1. Consultas por proximidad 5

2.2. Espacios vectoriales 6

2.3. Concepto de Índice Métrico 6

3. Trabajo previo **8**

3.1. Algoritmos basados en pivotes 8

3.2. Algoritmos basados en particiones compactas 9

3.3. Algoritmos basados en permutaciones 11

3.3.1. Índice invertido 12

III PROPUESTA **15**

4. Propuesta **16**

4.1. Propuesta 1 16

4.2. Propuesta 2 19

IV EXPERIMENTACIÓN **21**

5. Experimentación **22**

5.1. Bases de datos sintéticas 22

5.2. Bases de datos reales 35

V CONCLUSIONES **52**

6. Conclusiones **53**

Índice de figuras

3.1. Ejemplo de base de datos. 12

5.1. Dimensión 16 con $k = 16$, m_i={4,8,12}. 24
5.2. Dimensión 16 con $k = 32$, m_i={8,16,24}. 24
5.3. Dimensión 16 con $k = 64$, m_i={16,32,48}. 25
5.4. Dimensión 16 con $k = 128$, m_i={32,64,96}. 25
5.5. Dimensión 16 con $k = 256$, m_i={64,128,192}. 26
5.6. Dimensión 32 con $k = 16$, m_i={4,8,12}. 26
5.7. Dimensión 32 con $k = 32$, m_i={8,16,24}. 27
5.8. Dimensión 32 con $k = 64$, m_i={16,32,48}. 27
5.9. Dimensión 32 con $k = 128$, m_i={32,64,96}. 28
5.10. Dimensión 32 con $k = 256$, m_i={64,128,192}. 28
5.11. Dimensión 64 con $k = 16$, m_i={4,8,12}. 29
5.12. Dimensión 64 con $k = 32$, m_i={8,16,24}. 29
5.13. Dimensión 64 con $k = 64$, m_i={16,32,48}. 30
5.14. Dimensión 64 con $k = 128$, m_i={32,64,96}. 31
5.15. Dimensión 64 con $k = 256$, m_i={64,128,192}. 31
5.16. Dimensión 128 con $k = 16$, m_i={4,8,12}. 32
5.17. Dimensión 128 con $k = 32$, m_i={8,16,24}. 32
5.18. Dimensión 128 con $k = 64$, m_i={16,32,48}. 33
5.19. Dimensión 128 con $k = 128$, m_i={32,64,96}. 34
5.20. Dimensión 128 con $k = 256$, m_i={64,128,192}. 34

5.21. Variando m_s en Dim=16 36
5.22. Variando m_s en Dim=64 37
5.23. Valores elegidos 38
5.24. BD=colors, k = 16 y m_i={4,8,12} 39
5.25. BD=colors, k = 32 y m_i={8,16,24} 39
5.26. BD=colors, k = 64 y m_i={16,32,48} 40
5.27. BD=colors, k = 128 y m_i={32,64,96} 40
5.28. BD=NASA, k = 16 y m_i={8,16,24} 41
5.29. BD=NASA, k = 32 y m_i={8,16,24} 41
5.30. BD=NASA, k = 64 y m_i={16,32,48} 42
5.31. BD=NASA, k = 128 y m_i={32,64,96} 42
5.32. Variando m_s en BD=colors 43
5.33. Variando m_s en BD=NASA 44
5.34. Valores elegidos 45
5.35. BD=English, k = 16 y m_i={4,8,12} 45
5.36. BD=English, k = 32 y m_i={8,16,24} 46
5.37. BD=English, k = 64 y m_i={16,32,48} 46
5.38. BD=English, k = 128 y m_i={32,64,96} 47
5.39. BD=Spanish, k = 16 y m_i={4,8,12} 47
5.40. BD=Spanish, k = 32 y m_i={8,16,24} 48
5.41. BD=Spanish, k = 64 y m_i={16,32,48} 48
5.42. BD=Spanish, k = 128 y m_i={32,64,96} 49
5.43. Variando m_s en BD=English 50
5.44. Variando m_s en BD=Spanish 50
5.45. Valores elegidos 51

Índice de cuadros

3.1. Permutaciones 13
3.2. Índice invertido obtenido para el ejemplo. 13
3.3. $F'(u,q)$ 14

4.1. Nuevo índice invertido obtenido para el ejemplo. 16
4.2. $F^*(u,q)$ 19
4.3. $F'(u,q)$ 19

5.1. Valores de k, m_i y m_s en bases de datos sintéticas 23
5.2. Valores de k, m_i y m_s en bases de datos reales 38

Lista de algoritmos

1. Candidatos: nuevo índice 18
2. dPM(m_i,Π_q,Π_u) 18
3. Candidatos: m_s 20

Resumen

La búsqueda por similitud consiste en recuperar de una base de datos los elementos más parecidos a una consulta dada. Este tipo de búsquedas son indispensables en bases de datos multimedia donde una búsqueda secuencial resulta costosa. Una manera de resolver el problema es modelarlo como un espacio métrico, donde se requiere una función de distancia que evalúe la similitud entre los elementos. Los algoritmos que resuelven consultas por similitud en espacios métricos, suelen emplear índices para conocer la respuesta sin tener que comparar toda la base de datos. De entre las técnicas existentes, se encuentra la familia de algoritmos basados en permutaciones, en la que se elige un conjunto de elementos, llamados *permutantes*, que permiten ordenar el resto de elementos de la base de datos.

Este trabajo se enfocó en emplear dos técnicas que hacen uso de un índice invertido para una recuperación eficiente en este tipo de consultas. La primera propuesta modifica tanto la estructura del índice invertido como la distancia entre permutantes, presentadas en [AS10]. Mientras que la segunda propuesta sólo cambia el cálculo entre permutantes.

Abstract

Similarity searching consists of retrieving from a database the elements most similar to a given query. Such searches are indispensable in multimedia databases where a sequential search is costly. One way to solve the problem is to model it as a metric space, where a distance function is required to evaluate the similarity between the elements. Algorithms that solve similarity queries in metric spaces often use indices to know the answer without having to compare the entire database. Among the existing techniques, there is the family of algorithms based on permutations, in which a set of elements is chosen, called permutants, that allow to order the rest of the elements of the database.

This work focused on employing two techniques that make use of an inverted index for efficient recovery in this type of queries. The first proposal modifies both the structure of the inverted index and the distance between permutants, presented in [AS10]. While the second proposal only changes the calculation between permutants.

Palabras clave: Espacio métrico, permutantes, índice invertido, bases de datos, consulta por similitud.

Parte I

INTRODUCCIÓN

Capítulo 1

Introducción

Con el avance de la tecnología y el incremento del uso de ésta, se produce un aumento en la información generada, la cual posteriormente es almacenada en bases de datos. Si se quiere realizar una consulta se revisará la base de datos, para ello se efectúa una búsqueda que dependerá del tipo de base de datos que se utilice; por ejemplo, para una bases de datos tradicional que utiliza el modelo relacional, es decir, que la información puede ser organizada en dominios específicos o registros con una clave, se emplea una búsqueda por igualdad de la clave (o una parte de ésta).

No obstante, en la actualidad la mayoría de los datos se encuentran almacenados en bases de datos multimedia, en las que no se puede separar por dominios. De manera que el concepto de búsqueda para estos casos, básicamente trata de encontrar objetos similares, es decir realizar una búsqueda por similitud. El cómo serán comparados dos objetos, *función de distancia*, dependerá del tipo de datos.

Es posible llevar a cabo una búsqueda de forma secuencial, pero si la base de datos usada se trata de miles o millones de objetos y/o la comparación entre ellos es costosa (en tiempo y/o recursos), no resulta práctica dicha búsqueda. Una alternativa ante esto son los índices. Debido a que los *índices* tratan de localizar rápidamente los elementos relevantes.

Las propuestas existentes para el manejo de la información multimedial están, en su mayoría, basadas en extraer la información importante de los objetos y representar dicha información utilizando vectores de dimensión alta. Lo cual es conocido como extracción de características. Se define la extracción de características δ como una transformación de un objeto *(Obj)* a un vector de múltiples dimensiones ($\mathbb{R}^m$), esto es, $\delta : Obj \rightarrow \mathbb{R}^m$, que son almacenados en las bases de datos *multidimensionales*.

Generalmente, los índices empleados aprovechan cada vector (llamados índices para espacios multidimensionales o vectoriales), además de que presentan un buen rendimiento en dimensiones bajas (<20). A medida que aumenta la dimensión el rendimiento disminuye rápida-

mente, esto es conocido como la *maldición de la dimensión* [CNBYM01].

1.1. Aplicaciones

Existen muchas aplicaciones que utilizan el concepto de búsqueda por similaridad. Entre ellas podemos mencionar reconocimiento de patrones, recuperación de información y biología computacional. A continuación se describirán brevemente [CNBYM99]:

- *Reconocimiento de patrones*. El propósito es analizar una escena en el mundo real tomada por un dispositivo y llegar a una descripción de la escena útil para completar alguna tarea.
  Básicamente el reconocimiento de patrones se realiza en tres etapas: la primera es la segmentación de los objetos útiles en la entrada. La siguiente etapa usualmente está destinada a eliminar el ruido de esos objetos que pudiera interferir en su reconocimiento. En la tercera etapa se lleva a cabo la clasificación del objeto en una o más categorías, siendo aquí donde se requiere la búsqueda por similaridad, debido a que los objetos más parecidos entre sí deberían tener la misma categoría.

- *Recuperación de información*. La operación central es buscar en registros no estructurados (o semi-estructurados) aquellos más semejantes a uno de consulta. Un ejemplo de registro no estructurado es un documento en lenguaje natural.
  Un buscador de internet es un ejemplo de recuperación de información. Generalmente el usuario proporciona un conjunto de palabras, que conforman una especie de *documento de consulta* y obtiene como respuesta un conjunto de documentos ordenados por su relevancia (o similaridad) para la consulta. Una forma de modelar un documento es verlo como un conjunto de palabras, donde los términos forman un vector y una medida de semejanza entre ellos es la distancia entre dos vectores.

- *Biología computacional*. En esta área se utilizan secuencias que definen objetos básicos, como proteínas, ADN, etc. Si estas secuencias son modeladas como texto, entonces el problema es llevado a una búsqueda de cadenas de texto similares.
  En este tipo de búsquedas es poco probable que exista una secuencia idéntica a otra. Dependiendo de lo que se quiera evaluar al comparar dos secuencias, se utilizan distintas medidas de similitud entre cadenas, por ejemplo éstas pueden considerar las probabilidades de mutaciones, la similaridad entre estructuras tridimensionales, etc.

Parte II

CONCEPTOS BÁSICOS

Capítulo 2

Conceptos básicos

En este capítulo se presentan algunos conceptos que serán de utilidad a lo largo de esta tesis. El primer concepto que se describe es el de espacio métrico. El cual consta de un conjunto de datos y una medida de semejanza entre los elementos. Debido a que la medida está estrechamente ligada a la aplicación y a las características que se quieran evaluar, no existe una definición general de ésta.

Un espacio métrico es definido como sigue. Sea $\mathbb{X}$ un universo de objetos válidos, donde el subconjunto $\mathbb{U} \subseteq \mathbb{X}$ es una base de datos de tamaño n=$\mid \mathbb{U} \mid$, y una función de distancia con valor real no negativo $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}^+ \cup \{0\}$ definida entre los elementos de $\mathbb{X}$. Las funciones de distancia deben satisfacer las siguientes propiedades [Rud76]:

- Reflexivilidad: $d(x, x) = 0$
- Positividad estricta: $x \neq y \Longrightarrow d(x, y) > 0$
- Simetría: $d(x, y) = d(y, x)$
- Desigualdad del triángulo: $d(x, z) \leq d(x, y) + d(y, z)$

2.1. Consultas por proximidad

Los dos tipos de consultas por similitud se exponen a continuación, sea $q \in \mathbb{X}$ el elemento de la consulta, se tiene:

- **Consulta de rango**: recuperar los objetos dentro de una región centrada en un objeto de consulta dado q con radio r, formalmente, $(q, r)_d = \{u \in \mathbb{U} \mid d(u, q) \leq r\}$.

- **Consulta de los k vecinos más cercanos** $KNN(q)_d$: recuperar los k elementos en $\mathbb{U}$ más cercanos a q, es decir, sea $A \subseteq \mathbb{U}$ tal que $\forall u \in A$, $\forall v \in \mathbb{U} - A$, $d(q,u) \leq d(q,v)$, $k =\mid A \mid$.

El caso más común en muchas aplicaciones es cuando $k = 1$, es decir, se quiere recuperar al vecino más cercano (NN por sus siglas en inglés, *Nearest Neighbor*).

2.2. Espacios vectoriales

Un caso particular de los espacios métricos son los espacios vectoriales m-dimensionales, en los cuales los objetos son identificados con m coordenadas de valores reales, es decir, $(x_1, x_2, ..., x_m)$ y la distancia comúnmente usada es la métrica de Minkowski de orden p, d_p. Sean $u = (u_1, ..., u_m)$ y $q = (q_1, ..., q_m)$, la métrica d_p se define como:

$$d_p(u,q) = \sqrt[p]{\sum_{i=1}^{m} \mid u_i - q_i \mid^p}. \tag{2.1}$$

A continuación algunos de sus casos especiales:

Manhattan

$$d_1(u,q) = \sum_{i=1}^{m} \mid u_i - q_i \mid. \tag{2.2}$$

Euclidiana

$$d_2(u,q) = \sqrt[2]{\sum_{i=1}^{m} \mid u_i - q_i \mid^2}. \tag{2.3}$$

Máxima

$$d_\infty(u,q) = \max_{i=1,...,m} \mid u_i - q_i \mid. \tag{2.4}$$

En los espacios vectoriales los métodos usados suelen conocerse como *Spatial Access Methods (SAM)* [BBK01b]. Entre los más populares se encuentran *KD-Tree* [Ben75, Ben79], *R-Tree* [Gut84] y *X-Tree* [BKK96], por mencionar algunos [HS00, BBK+01a]. En general, dichas técnicas usan exhaustivamente la información de las coordenadas para agrupar y clasificar los objetos en el espacio.

2.3. Concepto de Índice Métrico

En espacios métricos los algoritmos que resuelven consultas por proximidad suelen emplear índices para conocer la respuesta sin tener que comparar toda la base de datos. De manera

que un índice puede verse como una estructura de datos que facilita la búsqueda en un conjunto de elementos.

Los algoritmos que resuelven consultas por proximidad generalmente tienen dos fases: la de pre-procesamiento y la de consulta. En la primera fase, se proyecta la base de datos en un nuevo espacio que permite crear un índice, que posteriormente se almacena. Durante la fase de consulta se recorre el índice para conocer la *lista de candidatos* que serán comparados directamente contra la consulta (comparaciones externas) y así descartar los elementos que no son parte de la respuesta. En esta lista se encuentran elementos que no son relevantes para la consulta, pero que no pudieron ser distinguidos por el índice. El caso ideal es que todos los elementos de la lista sean relevantes para la consulta.

Capítulo 3

Trabajo previo

En este capítulo se describen algunos de los algoritmos para la búsqueda por similitud en espacios métricos.

Como se mencionó en el capítulo 2 los índices que son empleados por las técnicas existentes durante las consultas tienen el propósito de reducir el costo de la búsqueda (generalmente evitando comparaciones de distancia). Los índices son usados para recorrer con cierto orden los datos y poder tomar decisiones sobre ellos, tales como compararlos o descartarlos, cuando es posible. De forma general, el proceso que realizan las propuestas existentes para la construcción de índices es utilizar ciertos elementos de la base de datos, precalcular y almacenar las distancias entre estos y el resto de la base de datos.

A continuación se describirán las tres familias de algoritmos en que se puede dividir el estado del arte [CNBYM01, HS03, CFN05]: los basados en pivotes, los basados en particiones compactas y los basados en permutantes.

3.1. Algoritmos basados en pivotes

La idea general de este tipo de algoritmos es elegir un conjunto $\mathbb{P} = \{p_1, p_2, ..., p_k\} \subseteq \mathbb{U}$, de tamaño $k = \mid \mathbb{P} \mid$, estos elementos se denominan *pivotes*. Para cada uno de los objetos de la bases de datos se precalculan las distancias hacia los pivotes en $\mathbb{P}$. El conjunto de las distancias resultantes $\{d(p_1,u), d(p_2,u), ..., d(p_k,u)\}$, para todo $u \in \mathbb{U}$, formará el índice.

Dada una consulta q, para cada $p_i \in \mathbb{P}$ se calcula $d(p_i,q)$. Con esto se puede acotar la distancia 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* $\mid d(q,p) - d(p,u) \mid \leq d(q,u) \leq d(q,p) + d(p,u)$.

La forma en que los algoritmos basados en pivotes usan el índice es la siguiente: en una consulta por rango, usando el lema 1, es posible descartar todos los elementos de la base de datos $u \in \mathbb{U}$ tales que existe $p \in \mathbb{P}$, $| d(q,p) - d(p,u) | > r$, pues es claro que sí $| d(q,p) - d(p,u) | > r$, entonces $d(q,u) > r$. Los elementos que no son descartados se comparan directamente contra la consulta (comparaciones externas).

3.2. Algoritmos basados en particiones compactas

Para estos tipos de algoritmos se divide el espacio en zonas tan compactas como sea posible. Se selecciona un conjunto de objetos $p_1, ..., p_k \subseteq \mathbb{U}$ y se divide el espacio, es decir, se distribuye los demás elementos entre las k zonas pertenecientes a cada p_i. Estos p_i se denominan como *centros* de su zona. El índice se constituye por los centros, los elementos que pertenecen a cada zona y, en algunos casos, información extra sobre las distancias. Un criterio para la distribución, entre varios, es asignar cada u a la zona cuyo p_i sea el más cercano. Este procedimiento se realiza recursivamente en cada zona, hasta que no sea posible o deseable dividir el espacio.

Los algoritmos basados en particiones compactas se dividen de acuerdo al procedimiento de búsqueda que usan: los que emplean un radio de cobertura r_c, los que emplean hiperplanos, y los que utilizan ambos criterios. El radio de cobertura r_c se define como la distancia máxima del centro de la zona a los elementos en ella. El hiperplano es la frontera entre dos zonas cuando cada elemento se asigna a su centro más cercano.

Durante la búsqueda usando el criterio del radio de cobertura, es posible acotar la distancia entre la consulta y los elementos en la base de datos de la siguiente forma.

Lema 2 *Dados tres objetos $u, p \in \mathbb{U}$, $q \in \mathbb{X}$, si p es el centro de la zona a la que pertenece u, con radio de cobertura r_c, sabemos que $d(q,u)$ está acotada por* $\max\{d(q,p) - r_c, 0\} \leq d(q,u) \leq d(q,p) + r_c$.

Demostración Ambos límites se obtienen 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 se tiene:

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

esto implica que:

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

Por otro lado, sabemos que $d(q,u) \geq 0$, por lo tanto el límite inferior es:

$$\max\{d(q,p) - r_c, 0\} \leq d(q,u).$$

El límite superior lo obtenemos de la desigualdad del triángulo:

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

Durante una consulta por rango $(q,r)_d$, esta familia de algoritmos descarta aquellas zonas de centro p tales que $d(p,q) - r_c > r$, ya que usando el lema 2 se sabe que cualquier elemento u en esa zona cumple con $d(q,u) > r$. En las zonas que no se descartan se repite el proceso hasta que llega a zonas sin divisiones. En éstas los elementos deben revisarse secuencialmente. En una búsqueda usando el criterio de los hiperplanos, es posible establecer una cota para $d(q,u)$.

Lema 3 *Sea $u \in \mathbb{U}$ un 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* $\max\left\{\frac{d(q,p_1)-d(q,p_2)}{2}, 0\right\} \leq d(q,u)$.

Demostración De la desigualdad del triángulo sabemos que:

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

lo cual lleva a:

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

De la misma forma 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 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:

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

Utilizando sólo el criterio de hiperplanos no es posible determinar una cota superior, porque los objetos pueden estar muy cerca o muy lejos de p_1 y p_2. En particular, una zona p_j debe ser revisada si $\frac{d(q,p_j)-d(q,p_i)}{2} \leq r$, y q pertenece a la zona de p_i. Para poder decidir visitar o descartar las otras zonas de la base de datos, se repite el mismo cálculo.

3.3. Algoritmos basados en permutaciones

La idea principal es seleccionar un conjunto $\mathbb{P} = \{P_1, P_2, ..., P_k\} \subseteq \mathbb{U}$, $|\mathbb{P}| = k$, cuyos elementos son llamados *permutantes*. Para cada objeto en la base de datos $u \in \mathbb{U}$ se calcula su distancia a cada permutante en $\mathbb{P}$, $d(u,p)$, es decir, $D_q = \{d(q,p_1), \cdots, d(q,p_k)\}$. Entonces ordena los elementos por distancia creciente a u, es decir, genera su permutación (Π_u).

La hipótesis es que dos elementos idénticos deben tener la misma permutación, mientras dos objetos similares o cercanos deben tener permutaciones similares. De manera que el objetivo de esta técnica es identificar a los elementos más cercanos entre sí usando la semejanza entre las permutaciones.

Durante la fase de consulta, se calcula la permutación Π_q para la consulta q. Posteriormente, para conocer en qué orden revisar los elementos, se ordenan los $\Pi_u, u \in \mathbb{U}$ de mayor a menor similaridad con respecto a Π_q. Existen varios enfoques para medir qué tan similares son dos permutaciones, entre ellas se encuentra Spearman Rho, $S_p(\Pi_q, \Pi_u)$, la cual formalmente se define como:

$$S_p(\Pi_u, \Pi_q) = \sum_{i=1}^{k} | \Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i) |^2 . \tag{3.1}$$

Donde Π_q^{-1} y Π_u^{-1}, son las permutaciones inversas de q y u respectivamente. Una permutación inversa nos indica en qué posición se encuentra cada uno de los permutantes. Otra medida de similaridad es Spearman Footrule [DG77, FKS02], que se define a continuación:

$$F(\Pi_u, \Pi_q) = \sum_{i=1}^{k} | \Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i) | . \tag{3.2}$$

Una de las alternativas para encontrar las permutaciones más cercanas a la consulta es ordenar la base de datos respecto a los valores que se obtienen de usar alguna de las medidas de similaridad, de forma creciente, esto puede ser usando *QuickSort*.

3.3.1. Índice invertido

En [AS10], los autores proponen usar un índice invertido para encontrar la permutación más similar a la permutación de la consulta. El índice invertido, es una de las propuestas existentes sobre algoritmos basados en permutaciones que usa la métrica Spearman Footrule.

En el pre-procesamiento, se calcula la permutación para cada elemento en la base de datos. Durante esta fase aparece el parámetro $m_i \leq k$, de manera que las permutaciones se cortan hasta los primeros m_i elementos. En el índice invertido cada permutante $p \in \mathbb{P}$ mantiene una lista de pares, el objeto y posición (u, ψ). Habrá un par (u, ψ) para cada elemento u que tiene este permutante entre los primeros m_i elementos de Π_u y donde $\psi \leq m_i$ representa en qué posición se encuentra el permutante p dentro de Π_u. Formalmente, cada lista del índice invertido es definido de la forma siguiente:

$$\mathbb{I}_{p_j} = \{(u, \psi) \mid \Pi_u^{-1}(p_i) = \psi \leq m_i\} \tag{3.3}$$

En la Figura 3.1 se presenta un ejemplo de base de datos, con la que se ilustrará el procedimiento, donde se encuentran nueve objetos u_i (círculos en blanco con líneas), seis permutantes p_k (círculos en negro) y una consulta q (círculo en azul). El cuadro 3.1 contiene las permutaciones de cada uno de los elementos de la base de datos y la permutación de la consulta. Para este ejemplo se usará $m_i = 4$. En el cuadro 3.2 se muestra las listas de pares para cada permutante.

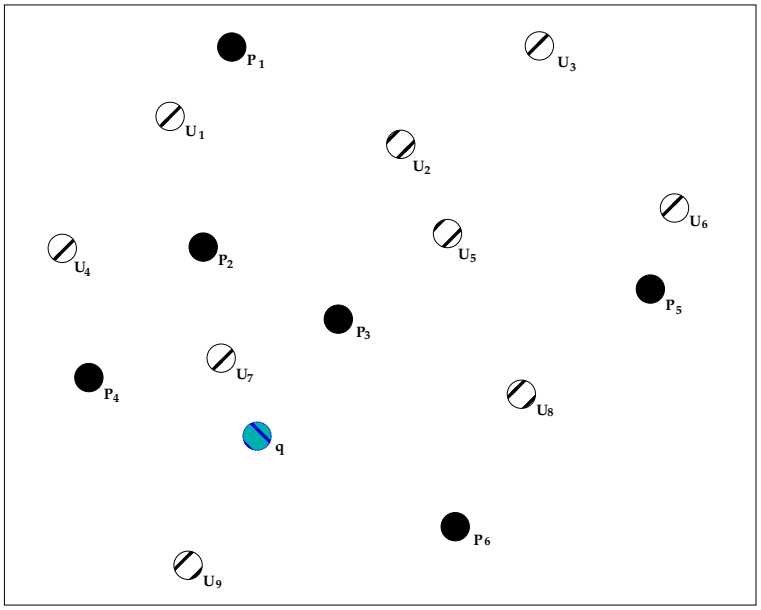


Figura 3.1: Ejemplo de base de datos.

En la fase de consulta se presenta otro parámetro, m_s, donde $m_s \leq m_i \leq k$. Los autores proponen mantener todos los elementos en la lista de los primeros m_s elementos de la permutación

Objeto	Π
u_1	1,2,3,4,5,6
u_2	3,1,2,5,6,4
u_3	5,1,3,2,6,4
u_4	4,2,1,3,6,5
u_5	3,5,2,1,6,4
u_6	5,3,6,2,1,4
u_7	2,3,4,1,6,5
u_8	6,5,3,2,1,4
u_9	4,6,3,2,5,1
q	3,2,4,6,1,5

Cuadro 3.1: Permutaciones

Permutante	(u,ψ)
P_1	(1,1),(2,2),(3,2),(4,3),(5,4),(7,4)
P_2	(1,2),(2,3),(3,4),(4,2),(5,3),(6,4),(7,1),(8,4),(9,4)
P_3	(1,3),(2,1),(3,3),(4,4),(5,1),(6,2),(7,2),(8,3),(9,3)
P_4	(1,4),(4,1),(7,3),(9,1)
P_5	(2,4),(3,1),(5,2),(6,1),(8,2)
P_6	(6,3),(8,1),(9,2)

Cuadro 3.2: Índice invertido obtenido para el ejemplo.

de la consulta, previamente calculada, es decir:

$$\mathbb{C} = \bigcup_{i=1}^{m_s} \mathbb{I}_p, p \in \Pi_q(i) \tag{3.4}$$

Como se mencionó antes, en esta etapa se calcula Spearman Footrule, y en caso de que esto no sea posible porque un permutante no se encuentra en el prefijo m_s de la permutación de la consulta, el autor propone utilizar $m_i + 1$. Se calcula $F'(u,q)$ para cada elemento en la base de datos:

$$F'(u,q) = \begin{cases} \mid \Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i) \mid, & \text{si} \quad \Pi_u^{-1}(p_i) \leq m_i \text{ y } \Pi_q^{-1}(p_i) \leq m_i \\ \\ m_i + 1, & \text{en otro caso} \end{cases} \tag{3.5}$$

El Cuadro 3.3 contiene el cálculo de $F'(u,q)$ para cada objeto de la base de datos. Como se mencionó en este ejemplo se usará $m_i = 4$, además de $m_s = 3$. De manera que el objeto más similar es u_7

Objeto	$F'(u,q)$
u_1	$\mid 1-3 \mid + \mid 2-2 \mid + \mid 3-4 \mid = 3$
u_2	$\mid 1-1 \mid + \mid 2-3 \mid + m_i + 1 = 6$
u_3	$\mid 1-3 \mid + \mid 2-4 \mid + m_i + 1 = 9$
u_4	$\mid 1-4 \mid + \mid 2-2 \mid + \mid 3-1 \mid = 5$
u_5	$\mid 1-1 \mid + \mid 2-3 \mid + m_i + 1 = 6$
u_6	$\mid 1-2 \mid + \mid 2-4 \mid + m_i + 1 = 8$
u_7	$\mid 1-2 \mid + \mid 2-1 \mid + \mid 3-3 \mid = 2$
u_8	$\mid 1-3 \mid + \mid 2-4 \mid + m_i + 1 = 9$
u_9	$\mid 1-3 \mid + \mid 2-4 \mid + \mid 3-1 \mid = 6$

Cuadro 3.3: $F'(u,q)$

Parte III

PROPUESTA

Capítulo 4

Propuesta

En [FRCI20] se presenta la base para este trabajo. A lo largo del capítulo se describe la propuesta, en la que se emplea la idea de índice invertido que fue descrito en el capítulo anterior. La propuesta se divide en dos técnicas, que se explicarán detalladamente a continuación.

4.1. Propuesta 1

La primera propuesta consiste en mantener un índice donde las listas contienen sólo aquellos objetos que tienen un permutante entre sus primeros m_i más cercanos. Formalmente el índice será de la siguiente forma: sea p_j un permutante, $p_j \in \mathbb{P}$, su lista en el índice invertido tendrá aquellos objetos $u \in \mathbb{U}$ tales que $\Pi_u^{-1}(p_j) \leq m_i$, es decir:

$$\mathbb{I}'_{p_j} = \{(u) \mid \Pi_u^{-1}(p_j) \leq m_i\}. \tag{4.1}$$

En el cuadro 4.1 se muestra el nuevo índice invertido, por la propuesta 1, para el ejemplo usado en el capítulo 3, véase el cuadro 3.2, recordando que $m_i = 4$.

Permutante	$\mathbb{I}'_{p_j}$
P_1	(1),(2),(3),(4),(5),(7)
P_2	(1),(2),(3),(4),(5),(6),(7),(8),(9)
P_3	(1),(2),(3),(4),(5),(6),(7),(8),(9)
P_4	(1),(4),(7),(9)
P_5	(2),(3),(5),(6),(8)
P_6	(6),(8),(9)

Cuadro 4.1: Nuevo índice invertido obtenido para el ejemplo.

Durante el tiempo de consulta, dada q, su permutación Π_q se calcula ordenando $D_q =$

$\{d(q,p_1),\cdots,d(q,p_k)\}$ de manera creciente. Se continúa utilizando el parámetro m_s en la búsqueda para acortar Π_q. En este caso, la lista de candidatos será la unión de las listas de cada permutante p de $\Pi_q^{-1}(p) \leq m_s$, las cuales se encuentran en el índice invertido. Formalmente:

$$\mathbb{C}' = \bigcup_{i=1}^{m_s} \mathbb{I}'_p, p \in \Pi_q(i). \tag{4.2}$$

Sin embargo, en este punto, D_q ya está calculada y se conoce la permutación completa Π_q de la consulta. La lista de candidatos para los vecinos más cercanos será $\mathbb{C}'$. Los candidatos serán ordenados por similitud entre permutaciones si se mantienen las permutaciones cortas de todos los elementos u de la base de datos ($\mid \Pi_u \mid = m_i$). Para esta propuesta, la ecuación (3.5), será modificada para evaluar la distancia entre permutaciones de diferente longitud, de la siguiente forma:

$$F^*(u,q)_{m_i} = \sum_{i=1}^{m_i} \mid i - \Pi_q^{-1}(\Pi_u(i)) \mid^2 . \tag{4.3}$$

Ésta manera de calcular la distancia entre permutantes nos permite evitar que algunos objetos queden fuera de la lista de candidatos debido a que éstos no se encuentren entre los elementos de las listas de permutantes. Por otro lado el valor *tope*, asignado al principio del proceso, nos brinda mayor control sobre los objetos que no pertenecerán a la lista de candidatos, ya que será el máximo valor que obtendrá un objeto que no se encuentre entre la unión de las lista de los permutantes. A continuación se define dicho valor:

$$tope = \frac{k*(k+1)}{2}. \tag{4.4}$$

Continuando con el ejemplo del capítulo anterior, ya que la permutación de q es $\Pi_q = \{3,2,4,6,1,5\}$, su permutación inversa es $\Pi_q^{-1} = \{5,2,1,3,6,4\}$, como tenemos $m_s = 3$ los elementos a usar son {3,2,4}. De modo que del índice invertido se toman las listas de los permutantes p_3, p_2 y p_4, que generarán $\mathbb{C}'$, para proceder a realizar el cálculo de $F^*(u,q)_{m_i}$.

Para el ejemplo usado, se obtiene como resultado de la unión de los permutantes $\mathbb{C}' = \{1,2,3,4,5,6,7,8,9\}$. En el cuadro 4.2 se muestran el cálculo de $F^*(u,q)_{m_i}$. Posteriormente se ordenan los resultados de forma creciente, de manera que el objeto más similar a la consulta es u_7, véase la Figura 3.1.

El algoritmo 1 muestra la forma en que son generados los candidatos para esta propuesta, empezando con la asignación del valor *tope* para cada elemento u, para posteriormente elegir las listas de cada permutante en el prefijo m_s de la permutación de la consulta y calcular la distancia $F^*(u,q)$. En el algoritmo 2 se presenta la forma en que se realiza el cálculo de la distancia entre permutaciones de la ecuación (4.3).

Algoritmo 1 Candidatos: nuevo índice

1: $tope = \frac{k*(k+1)}{2}$
2: Sea C el arreglo de candidatos
3: Sea n el tamaño de $\mathbb{U}$
4: **for** $i \in 1...n$ **do**
5: $C.lcs \leftarrow$tope
6: $C.i \leftarrow$i
7: $C.dist \leftarrow -1$
8: **end for**
9: **for** $i \in 1,...,m_s$ **do**
10: $perm \leftarrow \Pi_q(i)$
11: $lista \leftarrow$ m[perm] // m es el índice invertido
12: **while** $lista \neq NULL$ ***do***
13: ***if*** *C[lista->elemento].dist =-1* ***then***
14: $C[lista->elemento].lcs \leftarrow dPM(m_i, \Pi_q^{-1}, \Pi_u)$ *// u es lista->elemento*
15: $C[lista->elemento].dist \leftarrow 0$
16: ***end if***
17: $lista \leftarrow lista ->siguiente$
18: ***end while***
19: **end for**
20: *qsort(C)* *// QuickSort*
21: **return** C

Algoritmo 2 dPM(m_i,Π_q,Π_u)

1: **for** $i \in 1,...,m_i$ **do**
2: t$\leftarrow i$ - $\Pi_q(\Pi_u(i))$
3: $total$ +=t^2
4: **end for**
5: **return** $total$

Objeto	$F^*(u,q)_{m_i}$
u_1	$\mid 1-5 \mid^2 + \mid 2-2 \mid^2 + \mid 3-1 \mid^2 + \mid 4-3 \mid^2 = 21$
u_2	$\mid 1-1 \mid^2 + \mid 2-5 \mid^2 + \mid 3-2 \mid^2 + \mid 4-6 \mid^2 = 14$
u_3	$\mid 1-6 \mid^2 + \mid 2-5 \mid^2 + \mid 3-1 \mid^2 + \mid 4-2 \mid^2 = 42$
u_4	$\mid 1-3 \mid^2 + \mid 2-2 \mid^2 + \mid 3-5 \mid^2 + \mid 4-1 \mid^2 = 17$
u_5	$\mid 1-1 \mid^2 + \mid 2-6 \mid^2 + \mid 3-2 \mid^2 + \mid 4-5 \mid^2 = 18$
u_6	$\mid 1-6 \mid^2 + \mid 2-1 \mid^2 + \mid 3-4 \mid^2 + \mid 4-2 \mid^2 = 31$
u_7	$\mid 1-2 \mid^2 + \mid 2-1 \mid^2 + \mid 3-3 \mid^2 + \mid 4-5 \mid^2 = 3$
u_8	$\mid 1-4 \mid^2 + \mid 2-6 \mid^2 + \mid 3-1 \mid^2 + \mid 4-2 \mid^2 = 33$
u_9	$\mid 1-3 \mid^2 + \mid 2-4 \mid^2 + \mid 3-1 \mid^2 + \mid 4-2 \mid^2 = 16$

Cuadro 4.2: $F^*(u,q)$

4.2. Propuesta 2

Para la segunda propuesta, se mantiene la estructura del índice invertido que es propuesto en [AS10], pero se reemplazará el parámetro $m_i + 1$ por m_s en la ecuación (3.5). Formalmente:

$$F'(u,q) = \begin{cases} \mid \Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i) \mid, & \text{si} \quad \Pi_u^{-1}(p_i) \leq m_i \text{ y } \Pi_q^{-1}(p_i) \leq m_s \\ m_s, & \text{en otro caso} \end{cases} \tag{4.5}$$

En el Cuadro 4.3 se muestra el cálculo de $F'(u,q)$ para cada objeto en la base de datos, debido a que como resultado de la unión de los permutantes se tiene $\mathbb{C}' = \{1,2,3,4,5,6,7,8,9\}$. Como se mencionó se usará $m_s = 3$. Posteriormente se ordenan los resultados de forma creciente, en consecuencia el objeto más similar a la consulta es u_7.

Objeto	$F'(u,q)$
u_1	$\mid 1-3 \mid + \mid 2-2 \mid + \mid 3-4 \mid = 3$
u_2	$\mid 1-1 \mid + \mid 2-3 \mid + m_s = 4$
u_3	$\mid 1-3 \mid + \mid 2-4 \mid + m_s = 7$
u_4	$\mid 1-4 \mid + \mid 2-2 \mid + \mid 3-1 \mid = 5$
u_5	$\mid 1-1 \mid + \mid 2-3 \mid + m_s = 4$
u_6	$\mid 1-2 \mid + \mid 2-4 \mid + m_s = 6$
u_7	$\mid 1-2 \mid + \mid 2-1 \mid + \mid 3-3 \mid = 2$
u_8	$\mid 1-3 \mid + \mid 2-4 \mid + m_s = 7$
u_9	$\mid 1-3 \mid + \mid 2-4 \mid + \mid 3-1 \mid = 6$

Cuadro 4.3: $F'(u,q)$

En el algoritmo 3 se muestra la forma en que son generados los candidatos para la segunda propuesta. Se inicia con la asignación inicial de la distancia como m_s^2 para cada elemento u, para posteriormente elegir las listas de cada permutante del prefijo m_s de la consulta y calcular

la distancia correspondiente. Ésto último se encuentra en la línea 12, se realizó de tal forma debido a que se recorre la lista completa de cada permutante y a que se tienen dos casos, uno en el que el objeto u se encuentra en la lista y el caso contrario y así efectuar de manera correcta la ecuación (4.5).

Algoritmo 3 Candidatos: m_s

```
1:  Sea C el arreglo de candidatos
2:  Sea n el tamaño de 𝕌
3:  for i ∈ 1...n do
4:      C.lcs← m_s^2
5:      C.i←i
6:  end for
7:  for i ∈ 1,...,m_s do
8:      perm ← Π_q(i)
9:      lista ← m[perm]                 // m es el índice invertido
10:     while lista ≠ NULL do
11:         t ← |i-lista->posición|
12:         C[lista->elemento].lcs -= m_s-t
13:         lista ← lista ->siguiente
14:     end while
15: end for
16: qsort(C)                    // QuickSort
17: return C
```

Parte IV

EXPERIMENTACIÓN

Capítulo 5

Experimentación

En este capítulo se muestra el rendimiento de las propuestas experimentalmente. Para ello se probará con dos tipos de bases de datos: el primero son las bases de datos sintéticas, es decir, vectores aleatorios distribuidos de manera uniforme en el cubo unitario; el segundo son las bases de datos reales.

En las figuras que se presentan en esta sección se efectúa la comparación de los resultados para nuestras dos propuestas (denotamos Pm a la primera propuesta y Ps a la segunda), además de compararlos contra la técnica de permutantes (PP) y la que fue propuesta en [AS10] (Pi). Cabe mencionar que en las gráficas presentadas a lo largo de éste capítulo se denota como *Distancias* a la suma de las comparaciones internas y externas.

5.1. Bases de datos sintéticas

En las bases de datos sintéticas es posible controlar la dimensión de los objetos que componen las bases de datos, esto es diferente en las bases de datos reales. Una forma de controlar la dimensión del espacio es generar un conjunto uniformemente distribuido en el cubo unitario, y usar este conjunto como un espacio métrico abstracto.

Las dimensiones usadas para realizar las pruebas, fueron dim= 16, 32, 64, 128. Los valores que se utilizaron para k, m_i y m_s se muestran en el cuadro 5.1.

En primer lugar se presentan algunos de los resultados que se obtuvieron, de manera que se pueda observar el comportamiento que tienen las técnicas usadas, en cada una de las dimensiones mencionadas. Posteriormente se presentan las comparaciones con respecto al valor m_s.

En general, se puede notar menor cantidad de distancias calculadas cuando el número

k	m_i	m_s
16	4	1, 2, 3, 4
	8	2, 4, 6, 8
	12	3, 6, 9, 12
32	8	2, 4, 6, 8
	16	4, 8, 12, 16
	24	6, 12, 18, 24
64	16	4, 8, 12, 16
	32	8, 16, 24, 32
	48	12, 24, 36, 48
128	32	8, 16, 24, 32
	64	16, 32, 48, 64
	96	24, 48, 72, 96
256	64	16, 32, 48, 64
	128	32, 64, 96, 128
	192	48, 96, 144, 192

Cuadro 5.1: Valores de k, m_i y m_s en bases de datos sintéticas

de permutantes k es mayor, y los valores m_i y m_s son iguales.

Dimensión 16

En las Figuras 5.1, 5.2 se puede observar que Pm tiene un mejor rendimiento que las otras técnicas, por otro lado puede verse que PP y Pi se desempeñan mejor que nuestra segunda propuesta.

La Figura 5.1 corresponde a dimensión 16 cuando se usa $k = 16$, donde la primera columna agrupa a $m_i = 4$, la segunda a $m_i = 8$ y la tercera a $m_i = 12$. Para ver en detalle la diferencia entre los desempeños obtenidos se examina 5.2(b), donde $m_i = 8$ y $m_s = 2$, se tiene que para encontrar 8 vecinos más cercanos Pm ocupa 13691 distancias, mientras que Ps ocupa 56278 distancias, en el caso de PP y Pi se requieren 24675 y 23303 respectivamente, de igual manera para encontrar un vecino más cercano tenemos que Pm requieren 2054 distancias, Ps, PP y Pi, ocupan 16211, 7562 y 6850 respectivamente. En ambos casos se puede notar que la técnica que tiene el mejor desempeño es Pm, con una considerable diferencia. En 5.2(b) se muestra experimentalmente que es posible evitar calcular hasta un 44 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con la técnica PP.

En la Figura 5.2 se muestran resultados de la dimensión 16 cuando $k = 32$, en la cual se tiene $m_i = 8$ en la primera columna, $m_i = 16$ en la segunda y $m_i = 24$ en la tercera. De entre estas gráficas véase 5.2(e), cuyos valores son $m_i = 16$ y $m_s = 8$, en la cual las distancias requeridas para encontrar 8 vecinos más cercanos son 1874, 8794, 4382 y 5626 usando Pm, Ps, PP y Pi respectivamente. En este punto aún es notable la diferencia entre el desempeño de Pm con respecto a las otras técnicas. En 5.2(e) se muestra experimentalmente que es posible evitar calcular hasta un

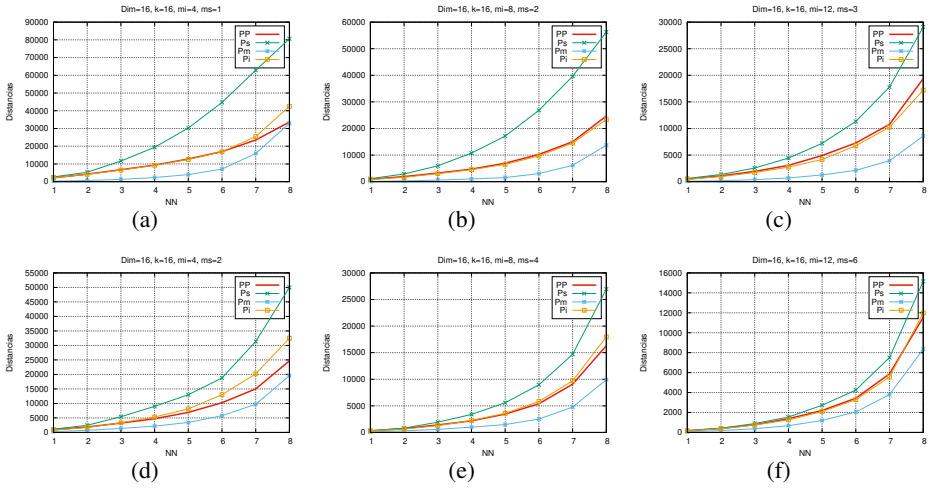


Figura 5.1: Dimensión 16 con $k = 16$, m_i={4,8,12}.

57 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con la técnica PP.

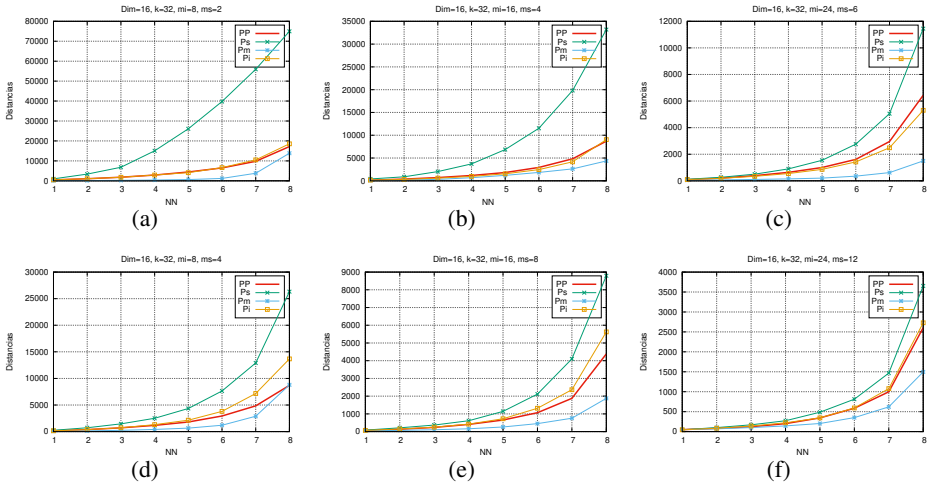


Figura 5.2: Dimensión 16 con $k = 32$, m_i={8,16,24}.

En la Figura 5.3 se agrupan resultados de dimensión 16 cuando $k = 64$, de manera que la primera columna contiene a $m_i = 16$, la segunda a $m_i = 32$ y la tercera a $m_i = 48$. Para este grupo de gráficas véase 5.3(a), se tiene que las distancias requeridas para encontrar 2 vecinos son 167, 1561, 229 y 185, que corresponden a Pm, Ps, PP y Pi respectivamente, mientras que para encontrar 7 vecinos Pm, Ps, PP, y Pi requieren 10246, 42515, 2637 y 2744, respectivamente. Nótese que el desempeño de Pm es mejor hasta encontrar 2 vecinos, al aumentar el número de vecinos a encontrar, también lo hace la cantidad de distancias. Un desempeño similar sucede en 5.3(d), en el resto su desempeño es distinto, en tanto que aumenta el número de vecinos disminuyen las distancias.

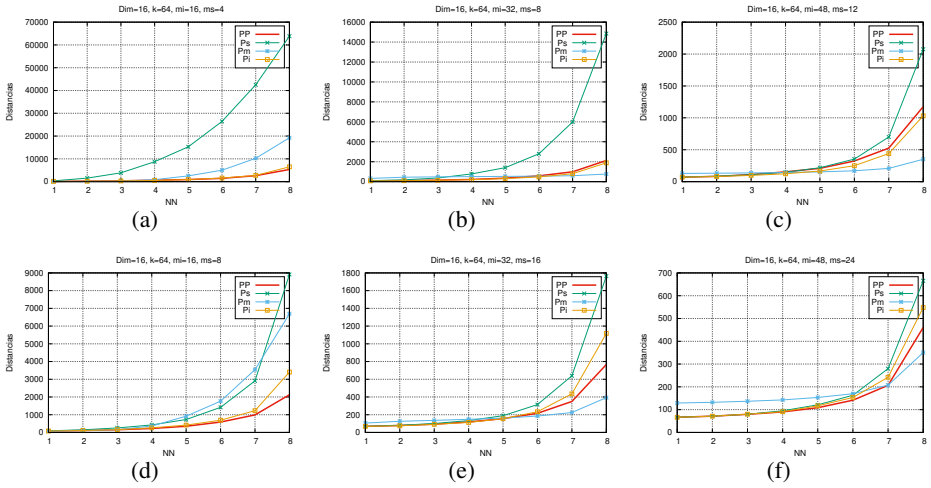


Figura 5.3: Dimensión 16 con $k = 64$, m_i={16,32,48}.

En la Figura 5.4 se muestran resultados de la dimensión 16 cuando $k = 128$, en la cual se tiene $m_i = 32$ en la primera columna, $m_i = 64$ en la segunda y $m_i = 96$ en la tercera. De entre estas gráficas véase 5.2(b), cuyos valores son $m_i = 64$ y $m_s = 16$, en la cual las distancias requeridas para encontrar 5 vecinos más cercanos son 289, 321, 192 y 184 usando Pm, Ps, PP y Pi respectivamente. En este caso la técnica que tiene un mejor desempeño es Pi.

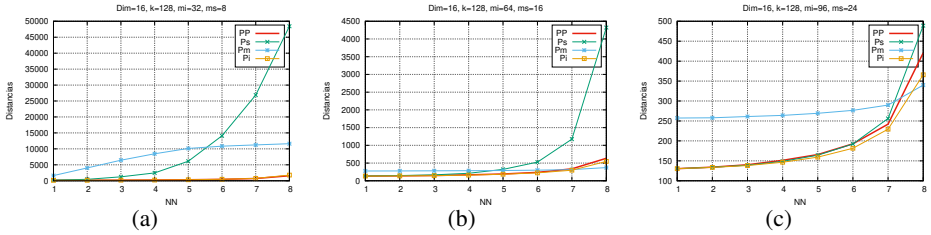
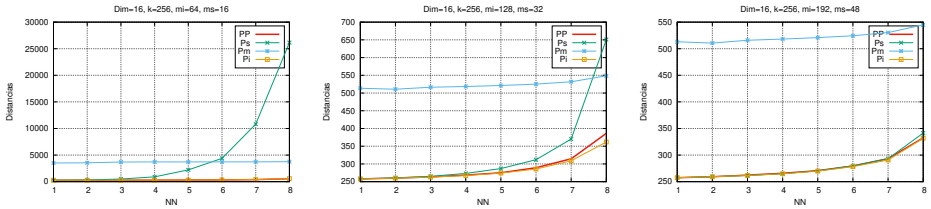


Figura 5.4: Dimensión 16 con $k = 128$, m_i={32,64,96}.

En la Figura 5.5 se muestran resultados de la dimensión 16 cuando $k = 256$, en la cual se tiene $m_i = 64$ en la primera columna, $m_i = 128$ en la segunda y $m_i = 192$ en la tercera. De entre estas gráficas véase 5.2(b), cuyos valores son $m_i = 128$ y $m_s = 32$, en la cual las distancias requeridas para encontrar 4 vecinos más cercanos son 518, 273, 268 y 267 usando Pm, Ps, PP y Pi respectivamente.

Dimensión 32

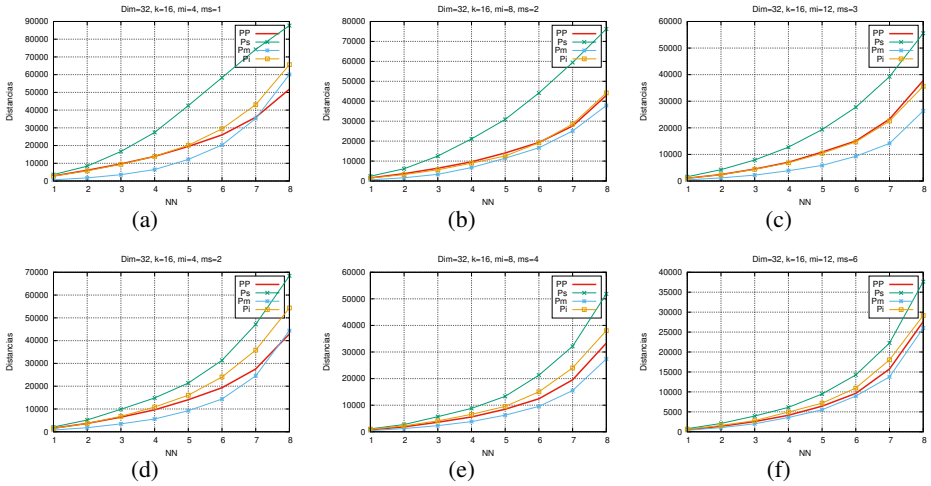
En las Figura 5.6 se presentan resultados de la dimensión 32 cuando $k = 16$, se tiene $m_i = 4$ en la primera columna, $m_i = 8$ en la segunda y $m_i = 12$ en la tercera. A fin de ver en detalle el rendimiento de las técnicas véase la gráfica 5.6(c) cuyos valores son $m_i = 12$ y $m_s = 3$, las distancias requeridas para encontrar 3 vecinos más cercanos son 2211, 7891, 4486 y 4410 para Pm, Ps, PP y



(a) (b) (c)

Figura 5.5: Dimensión 16 con $k = 256$, m_i={64,128,192}.

Pi. Nótese que a pesar de que se observa una disminución en la diferencia entre las técnicas, la que tiene mejor rendimiento es Pm. Otro aspecto a tener en cuenta se localiza en la primera columna, ya que se ve una disminución del rendimiento para encontrar 8 vecinos. En 5.6(c) se muestra experimentalmente que es posible evitar calcular hasta un 49 % de las distancias necesarias para encontrar 3 vecinos, esto comparado con la técnica Pi.



(a) (b) (c)

(d) (e) (f)

Figura 5.6: Dimensión 32 con $k = 16$, m_i={4,8,12}.

En la Figura 5.7 se agrupan resultados para la dimensión 32 cuando $k = 32$, en la primera columna se tiene $m_i = 8$, en la segunda $m_i = 16$ y en la tercera $m_i = 24$. Nótese que en este caso se tiene que para (c),(e) y (f) la técnica con mejor desempeño es Pm, mientras que para (a) Pm tiene mejor rendimiento hasta encontrar 3 vecinos, después es PP la que tiene mejor rendimiento, en (b) PP se desempeña mejor hasta encontrar 5 vecinos, para los vecinos siguientes Pm es la que tiene un mejor desempeño, por último en (d) Pm tiene mejor desempeño solo hasta encontrar 2 vecinos y posteriormente la técnica con mejor desempeño es PP.

En la Figura 5.8 se muestran resultados de la dimensión 32 cuando $k = 64$, se tiene $m_i = 16$ en la primera columna, $m_i = 32$ en la segunda y $m_i = 48$ en la tercera. Considérese la gráfica 5.8(b), las distancias requeridas para encontrar 3 vecinos son 368, 4724, 556 y 504 para

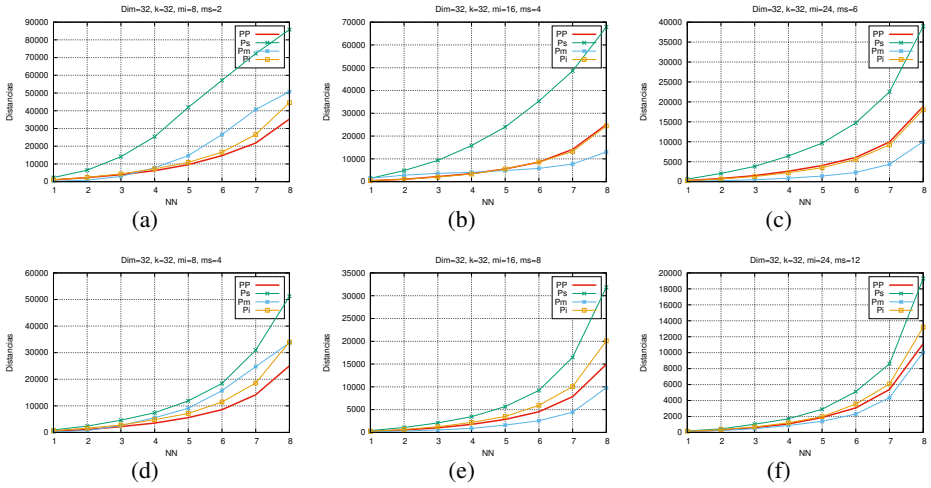


Figura 5.7: Dimensión 32 con $k = 32$, m_i={8,16,24}.

Pm, Ps, PP y Pi respectivamente. Véase que para la primer columna la técnica que tiene un mejor desempeño es PP, en las otras dos columnas nótese que sólo para encontrar hasta 2 vecinos el mejor rendimiento lo tiene la técnica PP, para los siguientes es Pm la que tiene mejor desempeño. En 5.8(b) se muestra que es posible evitar calcular un 29 % de las distancias necesarias para encontrar 3 vecinos, esto comparado con la técnica Pi.

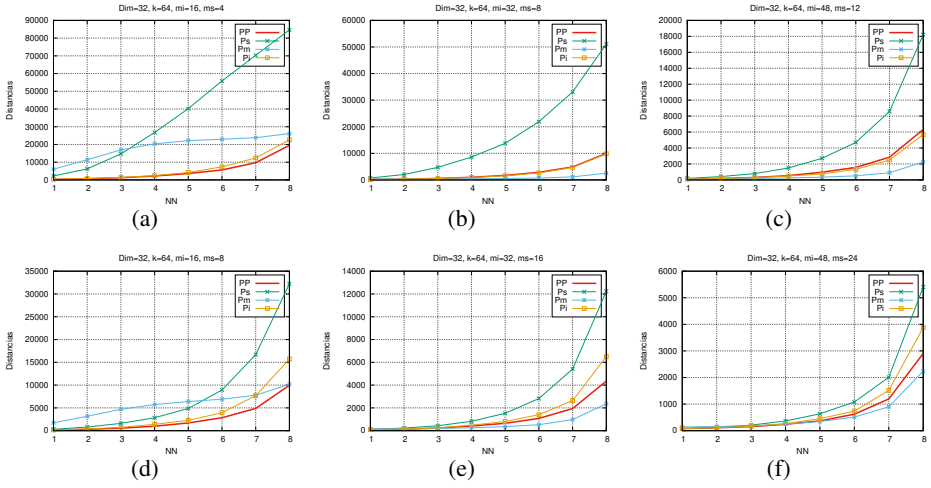
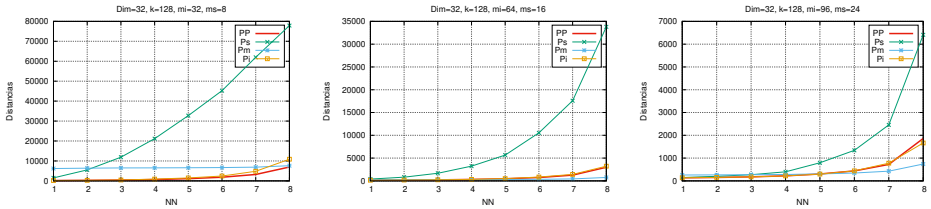


Figura 5.8: Dimensión 32 con $k = 64$, m_i={16,32,48}.

En la Figura 5.9 se presentan resultados de la dimensión 32 cuando $k = 128$, en la cual se tiene $m_i = 32$ en la primera columna, $m_i = 64$ en la segunda y $m_i = 96$ en la tercera. Considérese la gráfica 5.9(b) cuyos valores son $m_i = 64$ y $m_s = 16$, para Pm, Ps, PP y Pi las distancias que se requieren para encontrar 4 son 286, 3240, 317 y 328 respectivamente, nótese que a partir de este punto la técnica que se desempeña mejor es Pm, para los vecinos anteriores es la técnica PP, de

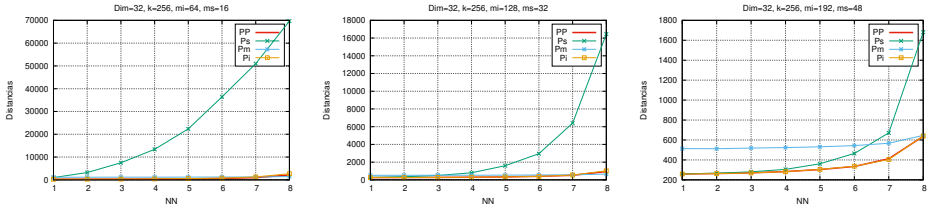
igual forma para la tercera columna. En cambio para la primera se puede observar que PP tiene un mejor desempeño.



(a) (b) (c)

Figura 5.9: Dimensión 32 con $k = 128$, m_i={32,64,96}.

En la Figura 5.10 se agrupan resultados de la dimensión 32 cuando $k = 256$, en la cual se tiene $m_i = 64$ en la primera columna, $m_i = 128$ en la segunda y $m_i = 192$ en la tercera. Para ver con mayor detalle el desempeño, considérese la gráfica 5.10(a), cuyos valores son $m_i = 64$ y $m_s = 16$, las distancias requeridas para encontrar 2 vecinos son 1152, 3211, 289 y 285, correspondientes a Pm, Ps, PP y Pi, para encontrar 8 vecinos se ocupan 1441, 69689, 2140 y 2700, Pm, Ps, PP y Pi respectivamente. En este caso la técnica que tiene un mejor desempeño es Pi, excepto para encontrar 8 vecinos.



(a) (b) (c)

Figura 5.10: Dimensión 32 con $k = 256$, m_i={64,128,192}.

Dimensión 64

La Figura 5.11 muestra resultados de la dimensión 64 cuando $k = 16$, la primera columna contiene $m_i = 4$, la segunda $m_i = 8$ y la tercera $m_i = 12$. Considérese la gráfica 5.11(b) cuyos valores son $m_i = 8$ y $m_s = 2$, las distancias requeridas para encontrar 1 vecino son 1764, 4277, 2743 y 2651, que corresponden a Pm, Ps, PP y Pi, mientras que para encontrar 6 vecinos se tienen 33110, 57163, 30282 y 31569, se puede observar que inicialmente la técnica con mejor desempeño es Pm, pero al aumentar el número de vecinos a encontrar cambia a PP. Sucede de la misma forma para (a) y (d), en cambio para (c), (e) y (f) es Pm la que tiene mejor rendimiento. En 5.11(b) se muestra que es posible evitar calcular un 33 % de las distancias necesarias para encontrar 1 vecino, esto comparado con la técnica Pi.

La Figura 5.12 agrupa resultados de la dimensión 64 cuando $k = 32$, en la primera

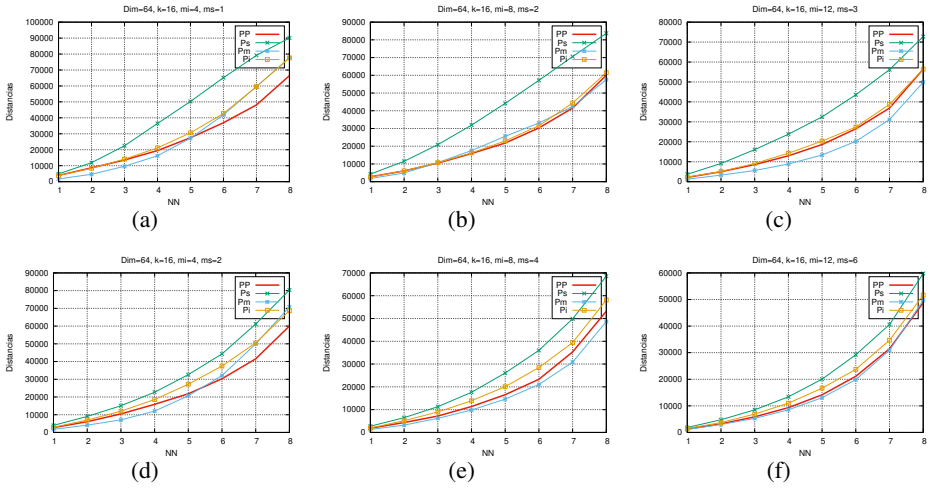


Figura 5.11: Dimensión 64 con $k = 16$, m_i={4,8,12}.

columna se localiza $m_i = 8$, en la segunda $m_i = 16$ y en la tercera $m_i = 24$. De entre estas gráficas elegimos 5.12(e), cuyos valores son $m_i = 24$ y $m_s = 12$, las distancias que se necesitan para encontrar 2 vecinos son 752, 1433, 896 y 1025, que corresponde a Pm, Ps, PP y Pi, para encontrar 8 vecinos son 27302, 42038, 29024 y 32830, Pm, Ps, PP y Pi respectivamente. A excepción de la primera columna, donde la técnica con mejor desempeño es PP, se tiene que Pm brinda mejores resultados. En 5.12(e) se muestra que es posible evitar calcular hasta un 5 % de las distancias necesarias para encontrar 2 vecinos, esto comparado con la técnica PP.

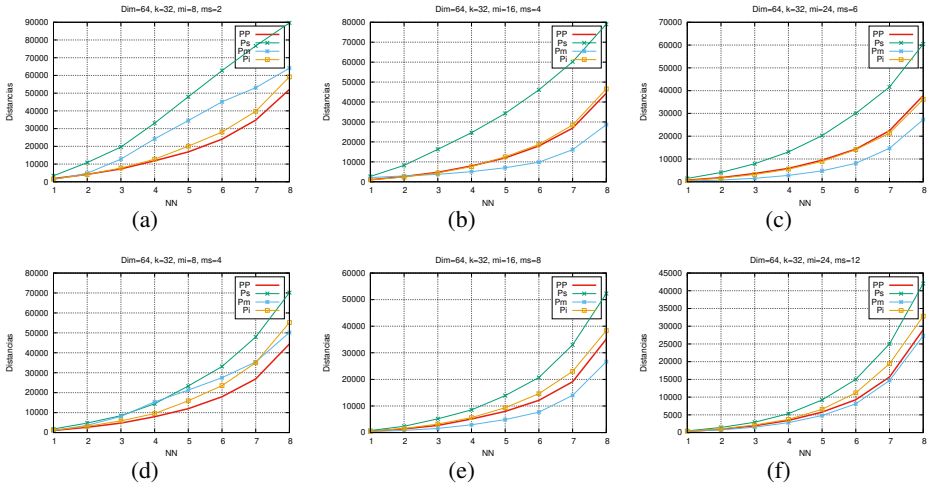
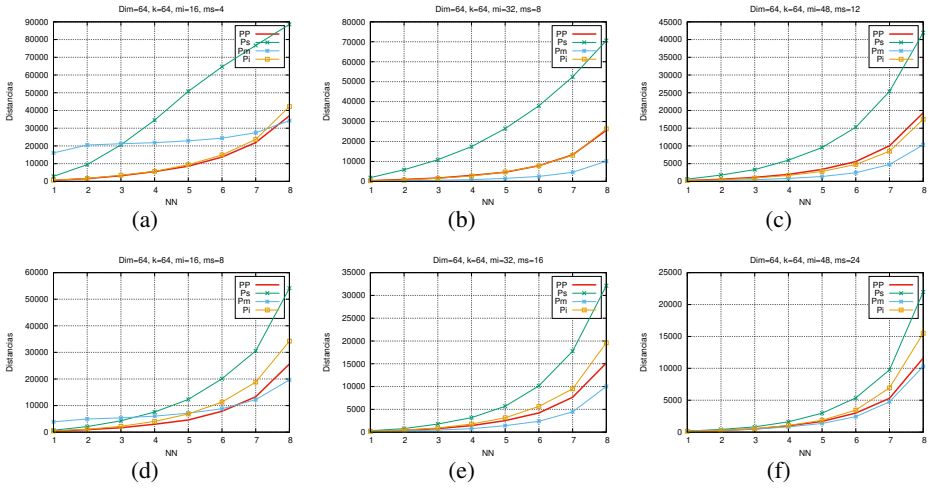


Figura 5.12: Dimensión 64 con $k = 32$, m_i={8,16,24}.

En la Figura 5.13 se muestran resultados de la dimensión 64 cuando $k = 64$, la primera columna contiene $m_i = 16$, la segunda 32 y la tercera $m_i = 48$. Considérese la gráfica 5.13(b) cuyos valores son $m_i = 32$ y $m_s = 8$, para este caso las distancias requeridas para encontrar 2 vecinos

cercanos son 358, 5846, 953 y 689, correspondientes a Pm, Ps, PP y Pi, mientras que para encontrar 8 vecinos se necesitan 10117, 70609, 25644 y 26362, Pm, Ps, PP y Pi. Se puede observar que la técnica con mejor rendimiento es Pm, al menos para los resultados de la segunda y tercera columna, para la primera fue PP. En 5.13(b) se muestra que es posible evitar calcular hasta un 60 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con la técnica PP.



(a) (b) (c)

(d) (e) (f)

Figura 5.13: Dimensión 64 con $k = 64$, m_i={16,32,48}.

En la Figura 5.14 se agrupan resultados de la dimensión 64 cuando $k = 128$, en la primera columna se tiene $m_i = 32$, en la segunda $m_i = 64$ y en la tercera $m_i = 96$, Para ver en detalle la diferencia entre las distancias véase la gráfica 5.14(e), cuyos valores son $m_i = 64$ y $m_s = 32$, para encontrar 3 vecinos Pm, Ps, PP y Pi requieren 322, 585, 289 y 343, respectivamente. Para encontrar 8 vecinos son requeridas 3183, 17108, 5748 y 8162 correspondientes a Pm, Ps, PP y Pi. La técnica que se desempeña mejor es Pm, excepto en 5.14(a) siendo PP la que tiene mejor desempeño hasta 6 vecinos. También nótese que en 5.14(d) en principio es PP la de mejor rendimiento y posteriormente es Pm. En 5.14(e) se muestra que es posible evitar calcular hasta un 44 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con la técnica PP.

La Figura 5.15 presenta resultados de la dimensión 64 cuando $k = 256$, $m_i = 64$ se localiza en la primera columna, $m_i = 128$ en la segunda y en la tercera $m_i = 192$. Considérese la gráfica 5.15(b) cuyos valores son $m_i = 128$ y $m_s = 32$, para encontrar 1 vecino las distancias necesarias son 514, 687, 266 y 264 , esto para Pm, Ps, PP y Pi, en cambio para encontrar 7 vecinos son 721, 29829, 1336 y 1421, correspondientes a Pm, Ps, PP y Pi. De manera que inicialmente Pi se desempeña mejor, al aumentar el número de vecinos la que se desempeña mejor es Pm, algo similar ocurre para los otros resultados. En 5.15(b) se muestra que es posible evitar calcular hasta un 46 % de las distancias necesarias para encontrar 7 vecinos, esto comparado con la técnica PP.

Dimensión 128

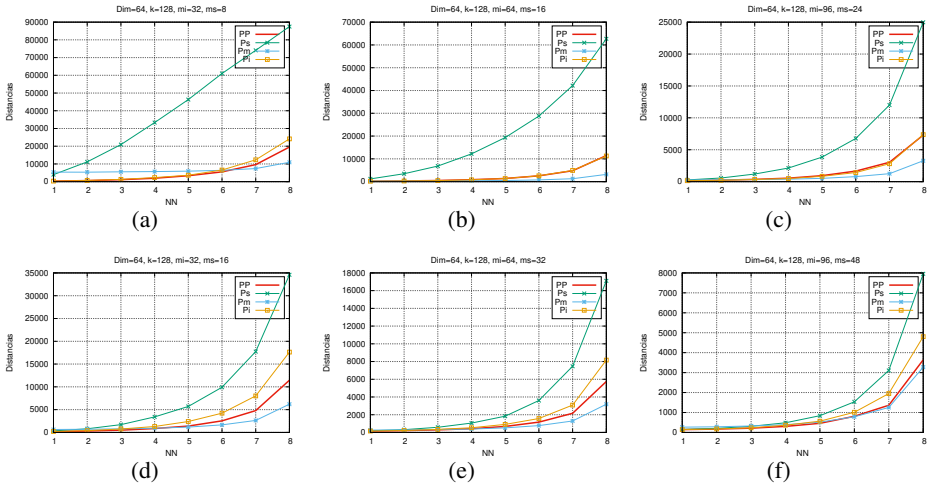


Figura 5.14: Dimensión 64 con $k = 128$, m_i={32,64,96}.

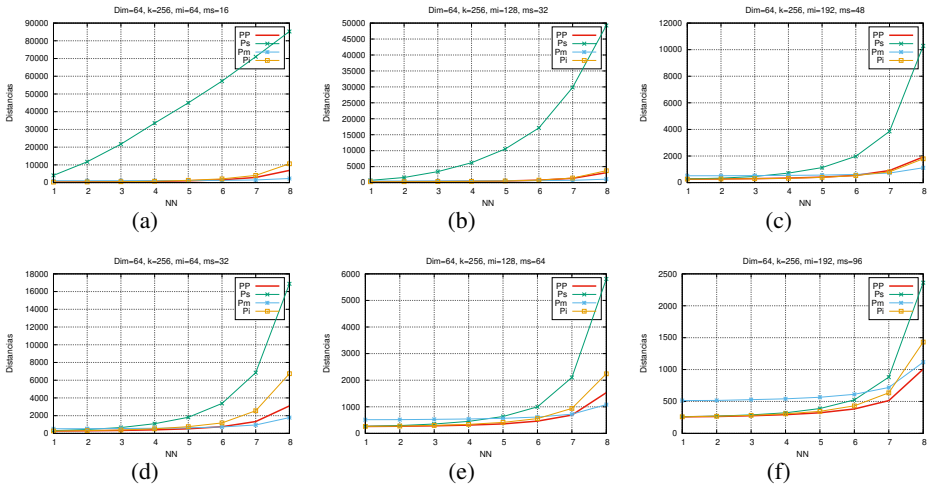


Figura 5.15: Dimensión 64 con $k = 256$, m_i={64,128,192}.

La Figura 5.16 muestra resultados para la dimensión 128 cuando $k = 16$, en la primera columna se localiza $m_i = 4$, en la segunda $m_i = 8$ y la tercera $m_i = 12$. Véase la gráfica 5.16(b) cuyos valores son $m_i = 8$ y $m_s = 2$, para encontrar 1 vecino las distancias necesarias son 3446, 5477, 4161 y 3538, correspondientes a Pm, Ps, PP y Pi. En tanto que para encontrar 4 vecinos son requeridas 25051, 36483, 22972 y 22810, Pm, Ps, PP y Pi, respectivamente. En este caso cambia la técnica con mejor desempeño, para algunos puntos es Pm, y para otros es PP, algo similar ocurre en 5.16(a) y 5.16(d), por el contrario en 5.16(c), 5.16(e) y 5.16(f) se mantiene con mejor desempeño la técnica Pm. En 5.16(b) se muestra que es posible evitar calcular un 2 % de las distancias necesarias para encontrar 1 vecino, esto comparado con la técnica PP.

En la Figura 5.17 agrupan resultados de la dimensión 128 cuando $k = 32$, la primera

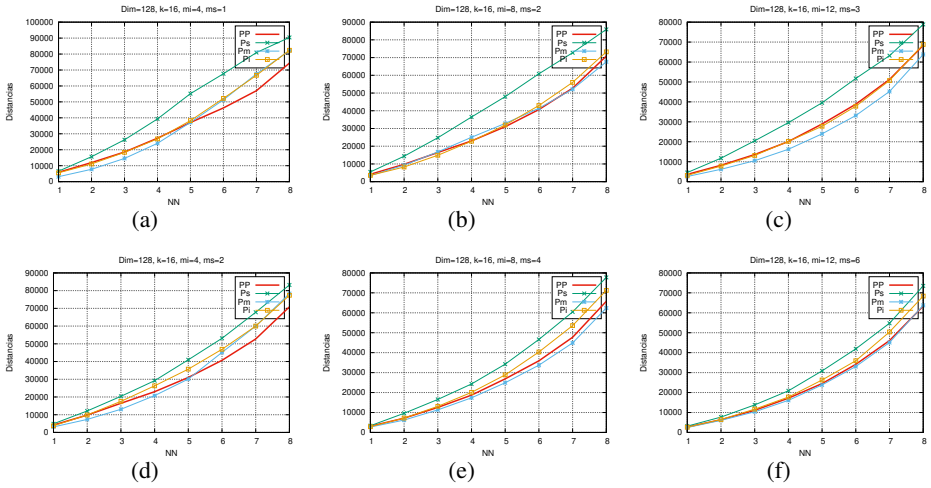


Figura 5.16: Dimensión 128 con $k = 16$, m_i={4,8,12}.

columna contiene a $m_i = 8$, la segunda a $m_i = 16$ y la tercera a $m_i = 24$. Considérese la gráfica 5.17(c) cuyos valores son $m_i = 24$ y $m_s = 6$, las distancias requeridas para encontrar 3 vecinos por Pm, Ps, PP y Pi son 5026, 14347, 7889 y 7339, mientras que para encontrar 7 vecinos las distancias necesarias son 29278, 53602, 37321 y 36847 correspondientes a Pm, Ps, PP y Pi. De manera que Pm tiene mejor desempeño, a excepción de la primera columna, donde puede observarse que es PP la que presenta un mejor desempeño. En 5.17(c) se muestra que es posible evitar calcular hasta un 20 % de las distancias necesarias para encontrar 7 vecinos, esto comparado con la técnica Pi.

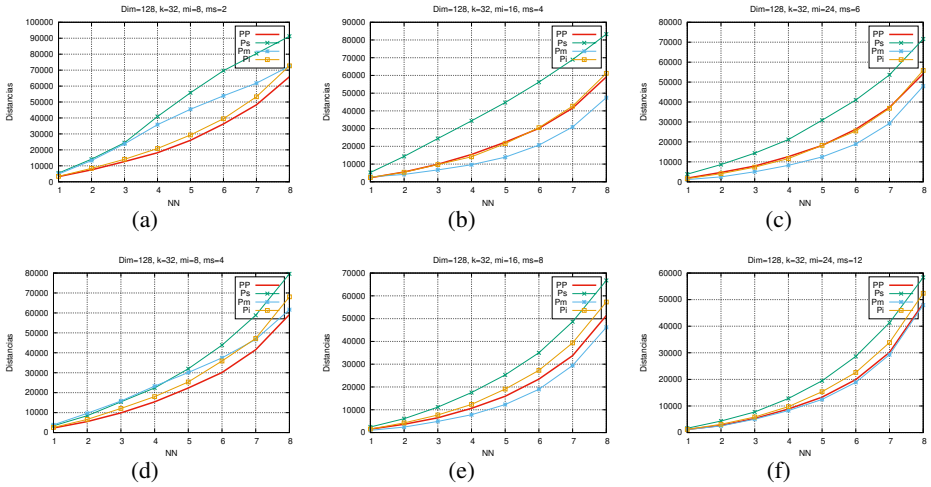


Figura 5.17: Dimensión 128 con $k = 32$, m_i={8,16,24}.

En la Figura 5.18 se presenta resultados de la dimensión 128 cuando $k = 64$, en la primera se tiene $m_i = 16$, en la segunda $m_i = 32$ y en la tercera $m_i = 48$. Para ver en detalle el rendimiento de las técnicas considérese la gráfica 5.18(b) cuyos valores son $m_i = 32$ y $m_s = 8$, las

distancias ocupadas por Pm, Ps, PP y Pi para encontrar 2 vecinos son 1002, 11664, 2818 y 2505, en el caso de 8 vecinos se necesitan 29034, 79682, 45340 y 47998 (Pm, Ps, PP y Pi). Es posible notar que Pm es la que se desempeña mejor, ocurre de la misma forma en 5.18(c), 5.18(e) y 5.18(f), en las gráficas restantes al inicio es PP y posteriormente Pm. En 5.18(b) se muestra que es posible evitar calcular hasta un 60 % de las distancias necesarias para encontrar 2 vecinos, esto comparado con la técnica Pi.

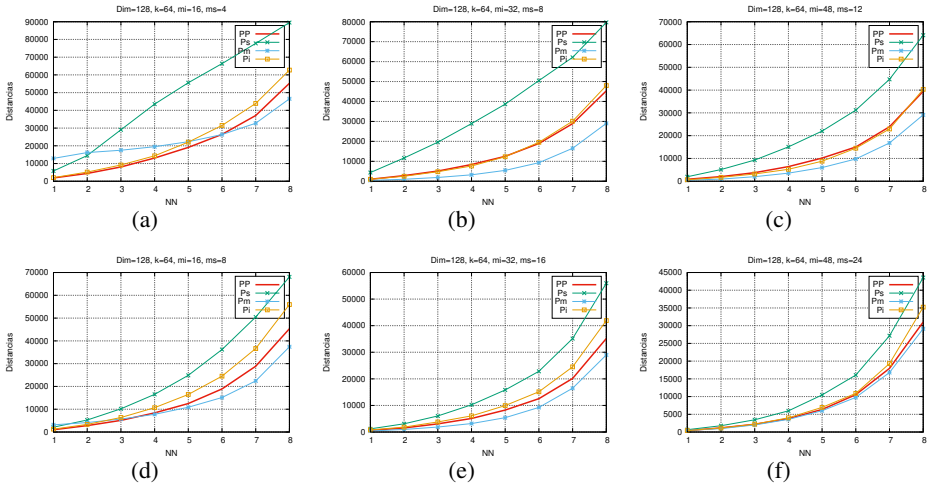


Figura 5.18: Dimensión 128 con $k = 64$, m_i={16,32,48}.

La Figura 5.19 muestra resultados de la dimensión 128 cuando $k = 128$, la primera columna incluye a $m_i = 32$, en la segunda a $m_i = 64$ y en la tercera a $m_i = 96$. De entre estas gráficas considérese 5.19(c) cuyos valores son $m_i = 96$ y $m_s = 24$, para encontrar 3 vecinos las distancias requeridas son 691, 4320, 1371 y 1043, que corresponden a Pm, Ps, PP y Pi, en el caso de 8 vecinos se tienen 11880, 48938, 20850 y 21612, en el mismo orden. La técnica con el mejor rendimiento es Pm, siendo de la misma forma para el resto de las gráficas a excepción de los dos primeros puntos en la gráfica 5.19(a), donde PP tiene un mejor rendimiento. En 5.19(c) se muestra que es posible evitar calcular hasta un 33 % de las distancias necesarias para encontrar 3 vecinos, esto comparado con la técnica Pi.

En la Figura 5.20 se agrupan resultados de la dimensión 128 cuando $k = 256$, en la primera columna se localiza $m_i = 64$, en ls segunda $m_i = 128$ y en la tercera $m_i = 192$. Considérese la gráfica 5.20(a) cuyos valores son $m_i = 64$ y $m_s = 16$, para encontrar 1 vecino las distancias requeridas por Pm, Ps, PP y Pi son 660, 6152, 427 y 436, respectivamente. Para el caso de 8 vecinos se ocupan 6912, 88482, 21395 y 28452. Obsérvese que PP tiene inicialmente un mejor rendimiento al aumentar el número de vecinos a encontrar la técnica que se desempeña mejor es Pm. Sucede de forma similar en las otras gráficas. En 5.20(a) se muestra que es posible evitar calcular hasta un 67 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con

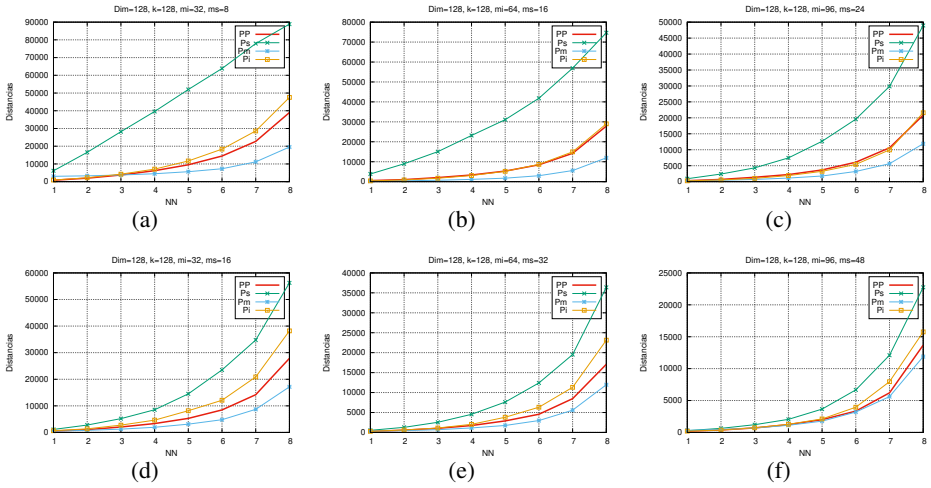


Figura 5.19: Dimensión 128 con $k = 128$, m_i={32,64,96}.

la técnica PP.

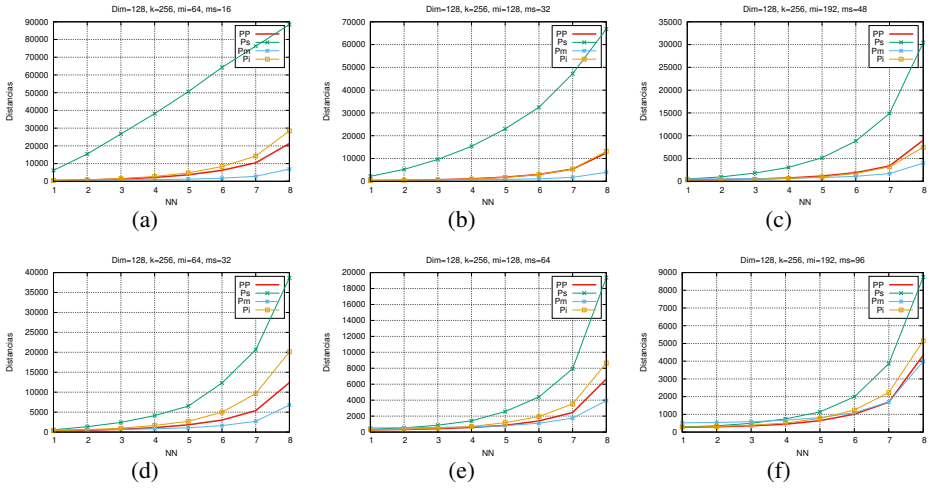


Figura 5.20: Dimensión 128 con $k = 256$, m_i={64,128,192}.

Variando el parámetro m_s

En esta sección se presentan las Figuras 5.21 y 5.22, en las cuales se realiza la comparación de los resultados con respecto al parámetro m_s, en la dimensión 16 y 64 respectivamente. Se muestran los 5 permutantes en cada dimensión, además de dos valores de m_i por cada permutante. Al igual que se observó en las gráficas anteriores, mientras más permutantes son usados menor es la cantidad de distancias que requieren las técnicas. Nótese que al aumentar el valor del parámetro m_s el número de distancias disminuye.

En la Figura 5.21 se agrupan las gráficas correspondientes a la dimensión 16. Como ya

se mencionó al aumentar k y m_s disminuyen las distancias requeridas. Asimismo puede observarse que los resultados de Pm, usando 16 y 32 permutantes muestran un mejor desempeño en comparación con los tres valores restantes, debido a que su rendimiento comienza a diminuir, es decir, requiere más distancias. Otro detalle notable se puede apreciar en los resultados de Pm cuyos m_i son los mayores (entre los dos valores que son presentados), ya que la diferencia entre un m_s y otro es mínima.

La Figura 5.22 agrupa las gráficas correspondientes a la dimensión 64. Similar a la dimensión 16, es posible ver que con el aumento de k y m_s las distancias requeridas disminuyen, excepto que en este caso los permutantes que muestran un mejor desempeño son 16,32 y 64. Al igual que en la dimensión anterior, se ve una mínima diferencia entre los m_s.

En la Figura 5.23 se agrupan los resultados elegidos para ambas dimensiones. Esto se generó a partir de la información obtenida de las gráficas que se encuentran en 5.21 y 5.22. De manera que se comparan las técnicas usando los parámetros que generan la menor cantidad de distancias. Se eligieron dos resultados por cada técnica.

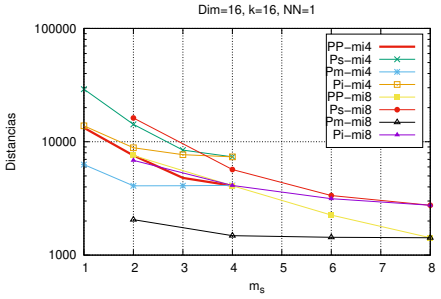
Es posible ver que en estas circunstancias la técnica Pm requiere mayor cantidad de distancias comparada con el resto de las técnicas. Se tiene que 5.23(a) pertenece a la dimensión 16, en este caso desde los primeros vecinos puede verse la diferencia entre las distancias requeridas por Pm y las otras técnicas, en tanto que para el último vecino es menor la cantidad de distancias comparado a un par de los resultados (Ps con valores $k = 64$, $m_i = 48$ y $m_s = 36$, Pi con valores $k = 64$, $m_i = 48$ y $m_s = 36$). En 5.23(b) se encuentran los resultados elegidos para la dimensión 64, para este caso obsérvese que las técnicas inicialmente tienen el mismo rendimiento, y conforme el número de vecinos a encontrar aumenta la diferencia entre las técnicas es más notoria, en especial respecto a Pm la cual requiere mayor cantidad de distancias. En ambas dimensiones puede verse que PP tiene un mejor desempeño.

5.2. Bases de datos reales

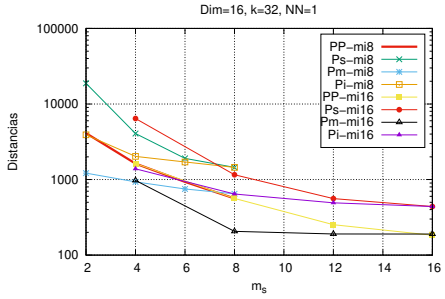
Dentro de las bases de datos reales utilizadas se encuentran *colors*, *NASA* y *diccionarios*. A continuación se describirán brevemente.

La base de datos denominada *colors*, consiste en un conjunto de 112,682 histogramas de colores, vectores de dimensión 112. Se utilizaron 112,182 para crear el índice y 500 para realizar las consultas. La base de datos denominada *NASA* consta de 40,150 vectores, de dimensión 20, fueron usados 39,650 elementos del conjunto para el índice y 500 para efectuar las consultas. La base de datos denominada *diccionarios*, se refiere a dos bases de datos de palabras, una es de español y la otra es de inglés, las cuales contienen 89,000 y 69,069 palabras respectivamente.

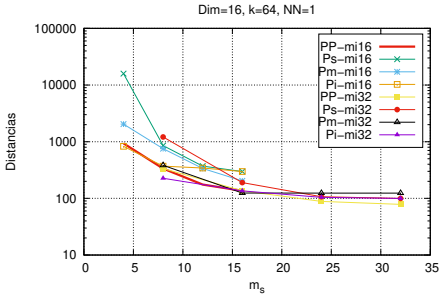
Para los experimentos el número de permutantes (k) y el valor de los parámetros m_i y



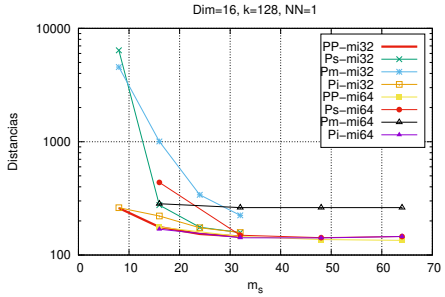
(a) $k = 16$, $m_i = \{4, 8\}$



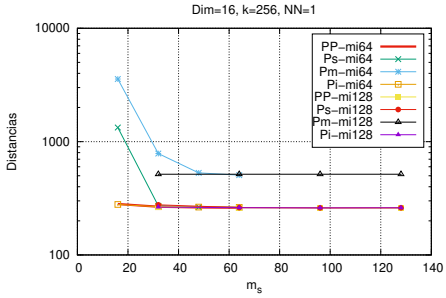
(b) $k = 32$, $m_i = \{8, 16\}$



(c) $k = 64$, $m_i = \{16, 32\}$

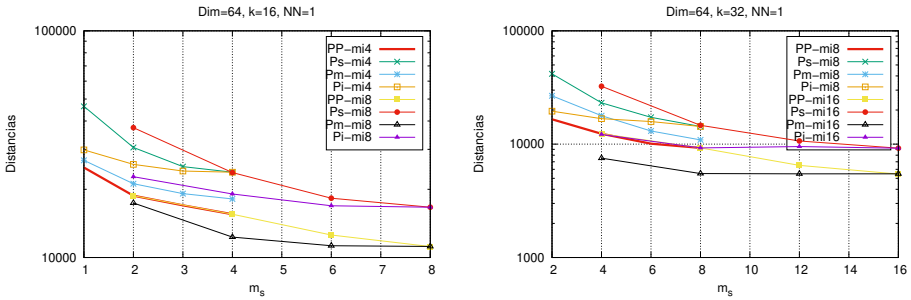


(d) $k = 128$, $m_i = \{32, 64\}$



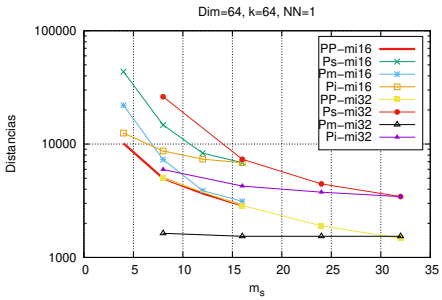
(e) $m_i = \{64, 128\}$

Figura 5.21: Variando m_s en Dim=16

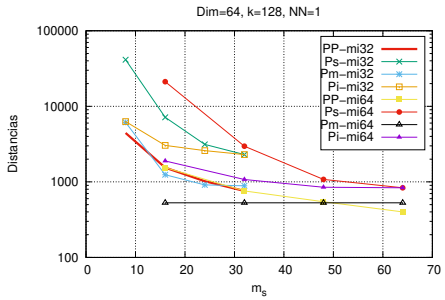


(a) $k = 16$, $m_i = \{4, 8\}$

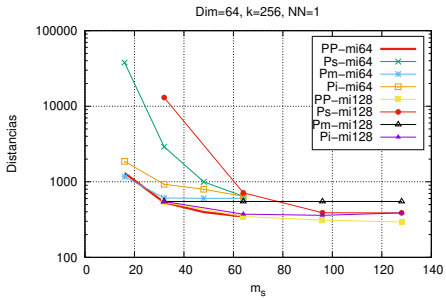
(b) $k = 32$, $m_i = \{8, 16\}$



(c) $k = 64$, $m_i = \{16, 32\}$



(d) $k = 128$, $m_i = \{32, 64\}$



(e) $k = 256$, $m_i = \{64, 128\}$

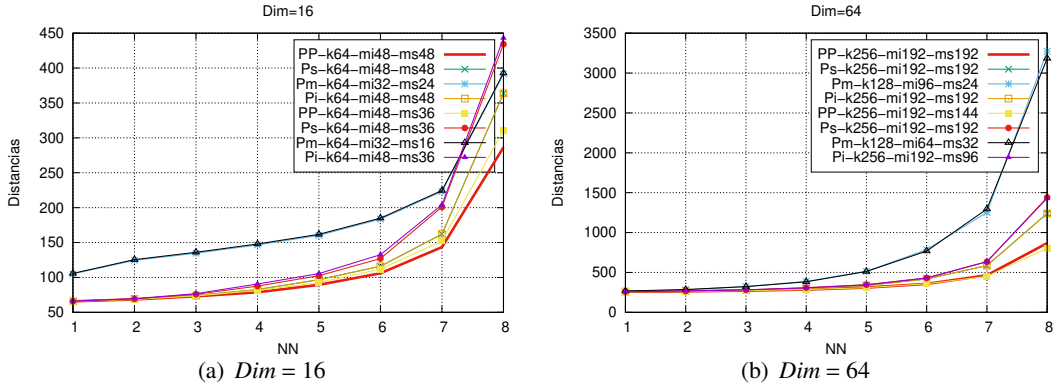
Figura 5.22: Variando m_s en Dim=64

m_s, se muestran en el cuadro 5.2.

De igual forma que se procedió con los resultados de las bases sintéticas, se hará con la bases de datos reales usadas.

Colors

En la Figura 5.24 se muestran resultados de la base de datos *Colors* cuando $k = 16$, la primera columna contiene a $m_i = 4$, la segunda a $m_i = 8$ y la tercera a $m_i = 12$. Considérese la gráfica 5.24(c) cuyos valores son $m_i = 12$ y $m_s = 3$, las distancias requeridas por Pm, Ps, PP y Pi para encontrar 8 vecinos son 21785, 49709, 47549 y 49106, respectivamente. Siendo así Pm la técnica que tiene mejor desempeño, de forma similar sucede en los otros casos. En 5.24(c) se muestra que es posible evitar calcular hasta un 54 % de las distancias necesarias para encontrar 8



(a) $Dim = 16$ (b) $Dim = 64$

Figura 5.23: Valores elegidos

k	m_i	m_s
16	4	1, 2, 3, 4
	8	2, 4, 6, 8
	12	3, 6, 9, 12
32	8	2, 4, 6, 8
	16	4, 8, 12, 16
	24	6, 12, 18, 24
64	16	4, 8, 12, 16
	32	8, 16, 24, 32
	48	12, 24, 36, 48
128	32	8, 16, 24, 32
	64	16, 32, 48, 64
	96	24, 48, 72, 96

Cuadro 5.2: Valores de k, m_i y m_s en bases de datos reales

vecinos, esto comparado con la técnica PP.

En la Figura 5.25 se presentan resultados de la base de datos *Colors* cuando $k = 32$, la primera columna tiene a $m_i = 8$, la segunda a $m_i = 16$ y la tercera a $m_i = 24$. De entre estas gráficas considérese 5.25(b) cuyos valores son $m_i = 16$ y $m_s = 4$, para este caso las distancias que se ocupan para encontrar 6 vecinos son 4513, 11947, 10886 y 11384 correspondientes a Pm, Ps, PP y Pi. Por lo que Pm mantiene el mejor rendimiento. En 5.25(b) se muestra que es posible evitar calcular hasta un 58 % de las distancias necesarias para encontrar 6 vecinos, esto comparado con la técnica PP.

La Figura 5.26 agrupa resultados de la base de datos *Colors* cuando $k = 64$, además la primera columna contiene a $m_i = 16$, la segunda a $m_i = 32$ y la tercera a $m_i = 48$. Para ver en detalle el rendimiento de las técnicas, véase la gráfica 5.26(a) cuyos valores son $m_i = 32$ y $m_s = 8$, las distancias que se requieren para encontrar 1 vecino son 193, 798, 509 y 534, correspondientes a Pm, Ps, PP y Pi. Para el caso de 4 vecinos se tiene 2333, 5168, 2170 y 2500 en el mismo orden que

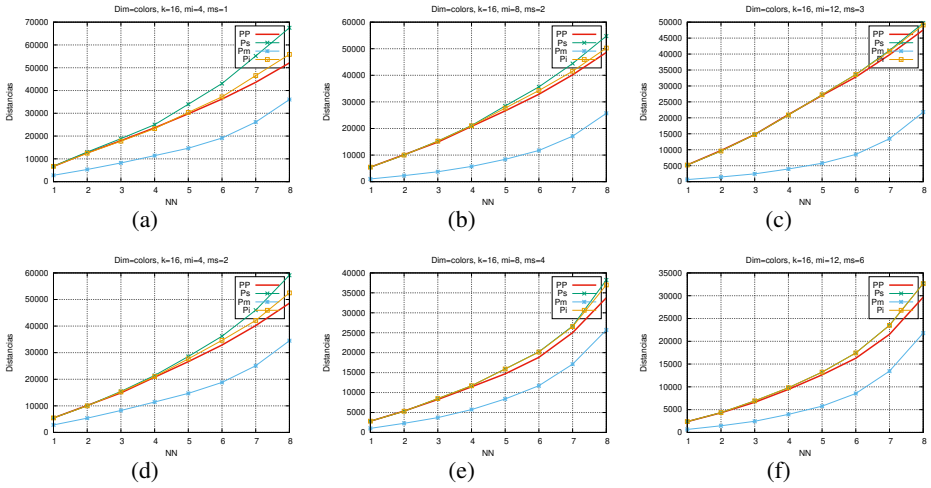


Figura 5.24: BD=colors, $k = 16$ y m_i={4,8,12}

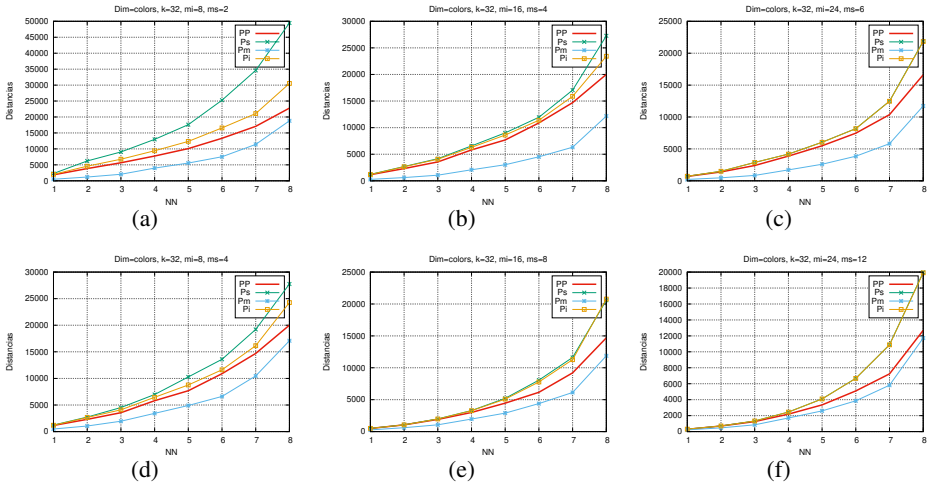


Figura 5.25: BD=colors, $k = 32$ y m_i={8,16,24}

el anterior. Nótese que para el primer caso es Pm la de mejor rendimiento pero en el segundo caso es PP. Es posible ver en algunas gráficas que la técnica con mejor desempeño cambia en varios puntos, alternando entre Pm, PP y Pi. En 5.26(a) se muestra que es posible evitar calcular hasta un 62 % de las distancias necesarias para encontrar 1 vecino, esto comparado con la técnica PP.

En la Figura 5.27 se muestran resultados para la base de datos *Colors* cuando $k = 128$, en la primera columna se localiza $m_i = 32$, en la segunda $m_i = 64$ y en la tercera $m_i = 96$. Véase la gráfica 5.27(c) cuyos valores son $m_i = 64$ y $m_s = 16$, para encontrar 1 vecino las distancias que ocupan Pm, Ps, PP y Pi son 239, 208, 229 y 208, respectivamente. Mientras que para encontrar 3 vecinos se ocupan 2286, 2219, 2170 y 2221, en el mismo orden. Aquí se puede observar algo similar a lo ocurrido en la figura 5.26, es decir, que la técnica con mejor desempeño varía, salvo

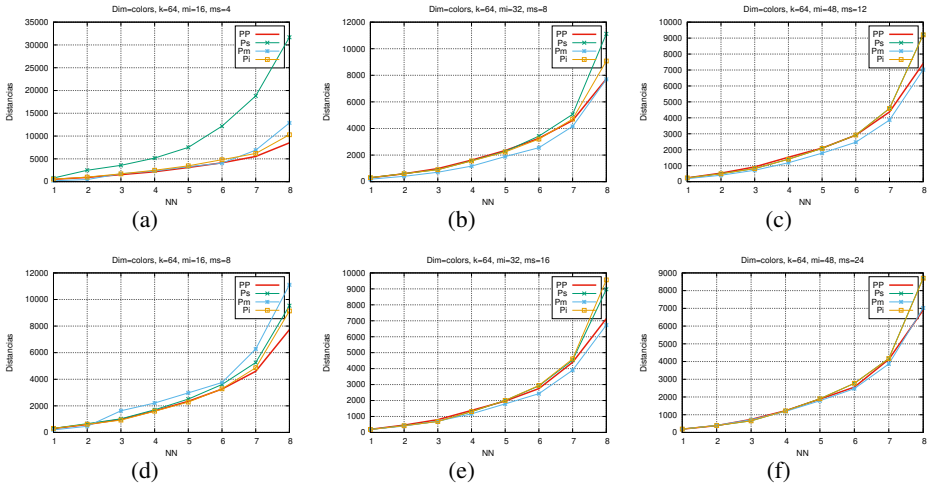


Figura 5.26: BD=colors, $k = 64$ y m_i={16,32,48}

que en este caso las técnicas son PP, Pi y Ps.

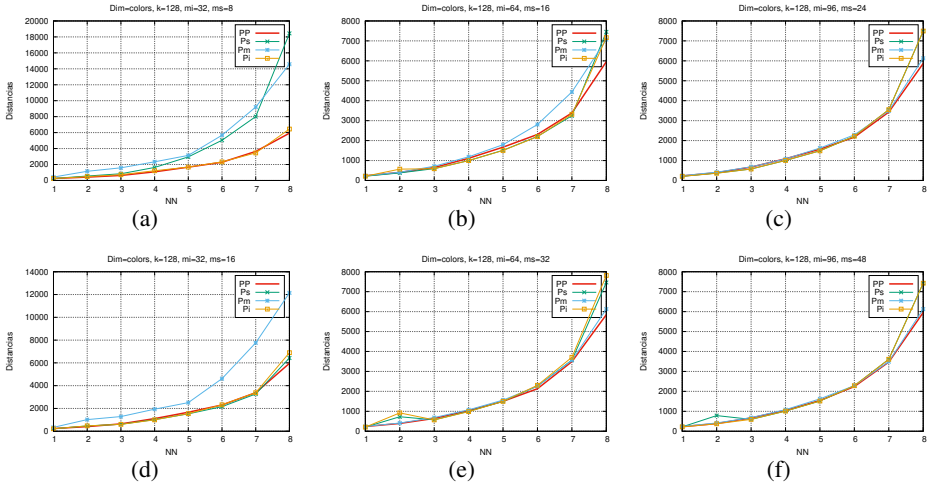


Figura 5.27: BD=colors, $k = 128$ y m_i={32,64,96}

NASA

En la Figura 5.28 se muestran resultados de la bases de datos *NASA* cuando $k = 16$, de manera que en la primera columna se agrupa $m_i = 4$, en la segunda $m_i = 8$ y en la tercera $m_i = 12$. De entre estas gráficas considérese 5.28(a) cuyos valores son $m_i = 4$ y $m_s = 1$, de manera que para encontrar 4 vecinos las distancias que se necesitan por Pm, Ps, PP y Pi son 316, 5057, 5020 y 5013. Al igual que en las otras gráficas, la técnica con mejor rendimiento es Pm.

La Figura 5.29 agrupan resultados para la base de datos *NASA* cuando $k = 32$, la primera columna contiene $m_i = 8$, la segunda a $m_i = 16$ y la tercera a $m_i = 24$. Considérese la gráfica

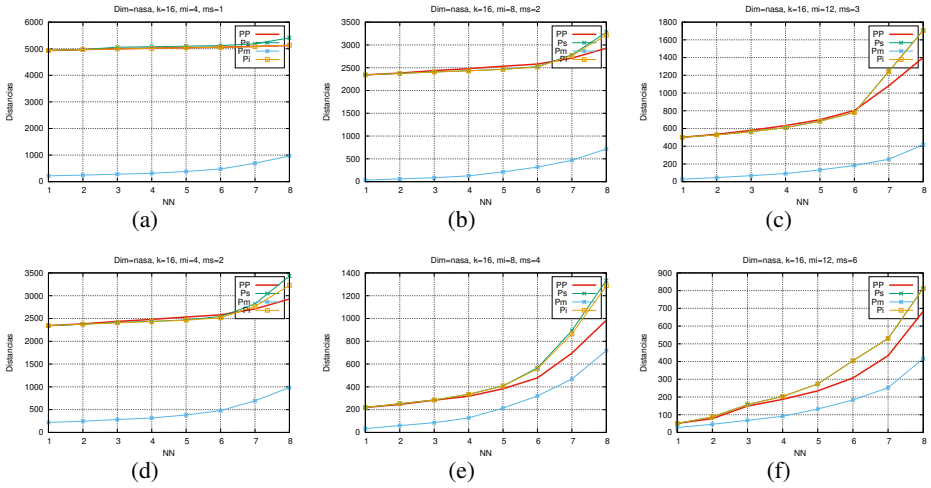


Figura 5.28: BD=NASA, $k = 16$ y m_i={8,16,24}

5.29(c) cuyos valores son $m_i = 24$ y $m_s = 6$, las distancias que requieren Pm, Ps, PP y Pi para encontrar 6 vecinos son 90, 254, 276 y 254. Nótese que aunque Pm sigue teniendo el mejor desempeño, pese a que la diferencia entre las técnicas disminuye. En 5.29(c) se muestra que es posible evitar calcular hasta un 64 % de las distancias necesarias para encontrar 6 vecinos, esto comparado con la técnica Pi.

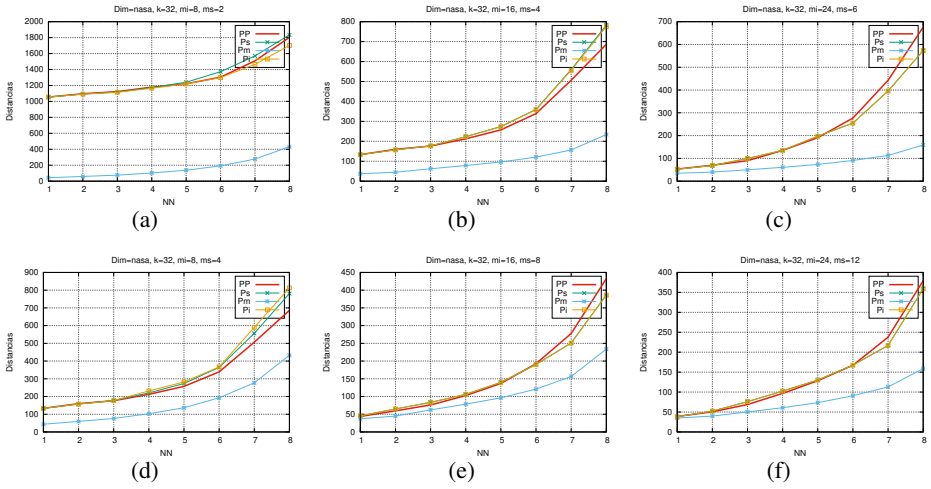
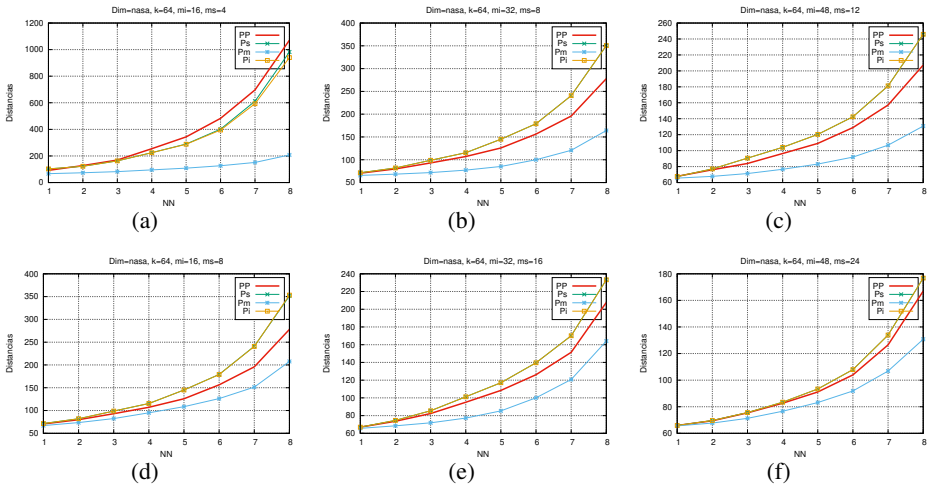


Figura 5.29: BD=NASA, $k = 32$ y m_i={8,16,24}

En la Figura 5.30 se presentan resultados de la base de datos *NASA* cuando $k = 64$, en la primera columna se localiza $m_i = 16$, en la segunda $m_i = 32$ y la tercera $m_i = 48$. Para ver el rendimiento considérese la gráfica 5.30(c) cuyos valores son $m_i = 48$ y $m_s = 12$, para encontrar 1 vecino las distancias que requieren Pm, Ps, PP y Pi son 65, 67, 67 y 67, en este caso la diferencia entre la mayor cantidad de distancias y la menor disminuye considerablemente, a mayor número de

vecinos a encontrar también aumenta la diferencia entre las distancias requeridas. En este caso aún se mantiene Pm como la técnica que tiene mejor rendimiento. En 5.30(c) se muestra que es posible evitar calcular hasta un 2 % de las distancias necesarias para encontrar 1 vecino, esto comparado con las técnicas Pp y Pi.

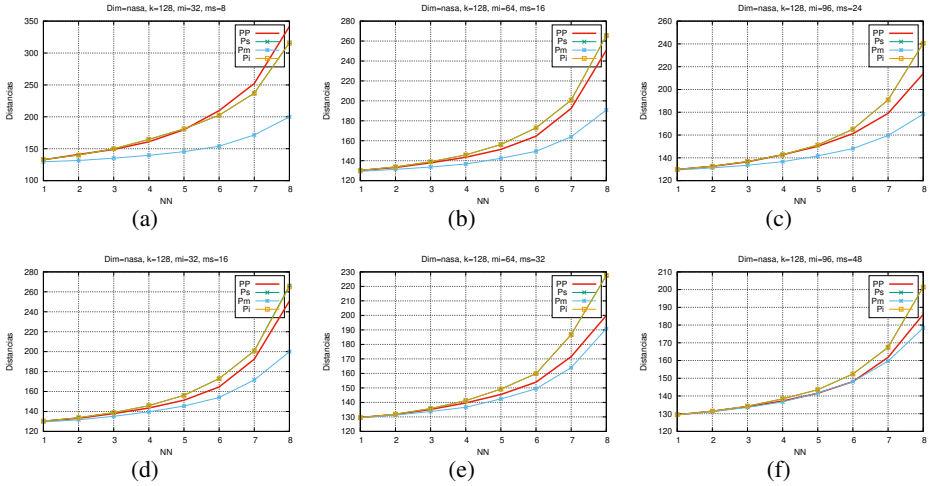


(a) (b) (c)

(d) (e) (f)

Figura 5.30: BD=NASA, $k = 64$ y m_i={16,32,48}

En la Figura 5.31 se muestran resultados de la base de datos *NASA* cuando $k = 128$, la primera columna se contiene a $m_i = 32$, la segunda a $m_i = 64$ y la tercera a $m_i = 96$. Considérese la gráfica 5.31(c) cuyos valores son $m_i = 96$ y $m_s = 24$, las distancias necesarias para encontrar 2 vecinos son 131, 132, 132 y 132, correspondientes a Pm, Ps, PP y Pi. Es posible observar que la diferencia entre la cantidad de distancias que requiere cada técnica disminuye, pese a ello Pm tiene mejor desempeño en la mayoría de puntos.



(a) (b) (c)

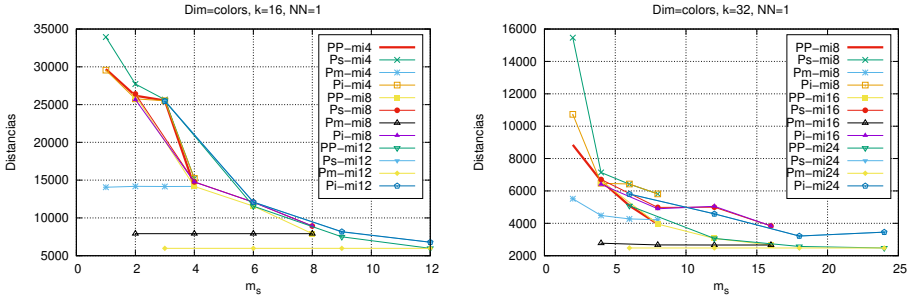
(d) (e) (f)

Figura 5.31: BD=NASA, $k = 128$ y m_i={32,64,96}

Variando m_s

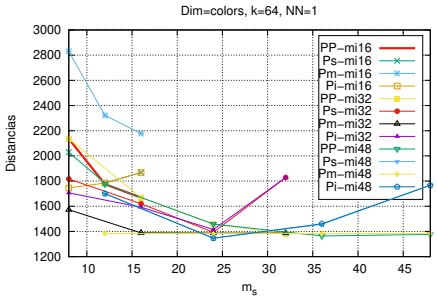
En esta sección se presentan las Figuras 5.32 y 5.33, en las cuales se realiza la comparación de los resultados con respecto al parámetro m_s, en las bases de datos *Colors* y *NASA* respectivamente. En ambas bases se muestran los 4 permutantes usados, además de los 3 valores de m_i por cada permutante, para encontrar 1 vecino más cercano.

En la Figura 5.32 se agrupan las gráficas correspondientes a la base de datos *Colors*. Nótese que en el caso de Pm algunos resultados se mantienen constantes, es especial cuando el valor de k es pequeño. También obsérvese que existen puntos donde coinciden las distancias requeridas, en particular Pm con PP y Ps con Pi, ambas cuando $m_s = m_i$.

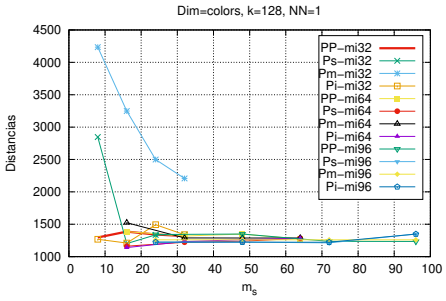


(a) $k = 16$, $m_i = \{4, 8, 12\}$

(b) $k = 32$, $m_i = \{8, 16, 24\}$



(c) $k = 64$, $m_i = \{16, 32, 48\}$

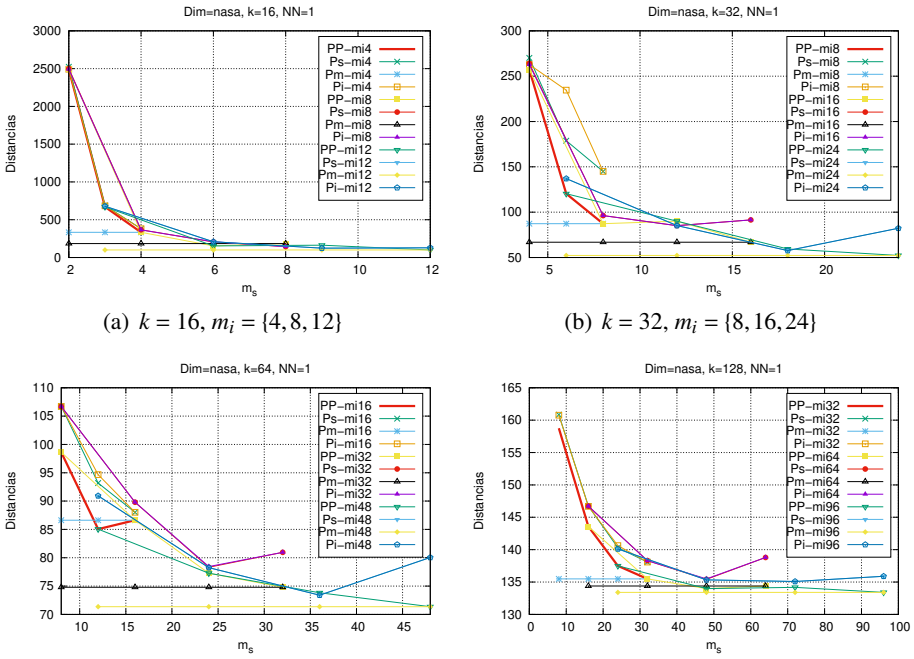


(d) $k = 128$, $m_i = \{32, 64, 96\}$

Figura 5.32: Variando m_s en BD=colors

En la Figura 5.32 se agrupan las gráficas correspondientes a la base de datos *NASA*. De forma similar que en la base de datos *Colors* Pm mantiene constante los resultados de cada m_i, a excepción que en este caso se percibe en los 4 valores de k usados. También se puede observar que existen puntos donde coinciden las distancias requeridas, excepto que en este caso se puede ver coincidencia en una mayor cantidad de puntos, al igual que en *Colors* es Pm con PP y Ps con Pi, en especial cuando $m_s = m_i$.

En la Figura 5.34 se agrupan los resultados elegidos para ambas bases de datos. Esto se generó a partir de las gráficas que se encuentran en 5.32 y 5.33. De modo que se comparan las técnicas usando los parámetros que generan la menor cantidad de distancias. Se eligieron dos



(a) $k = 16$, $m_i = \{4, 8, 12\}$ (b) $k = 32$, $m_i = \{8, 16, 24\}$

(c) $k = 64$, $m_i = \{16, 32, 48\}$ (d) $k = 128$, $m_i = \{32, 64, 96\}$

Figura 5.33: Variando m_s en BD=NASA

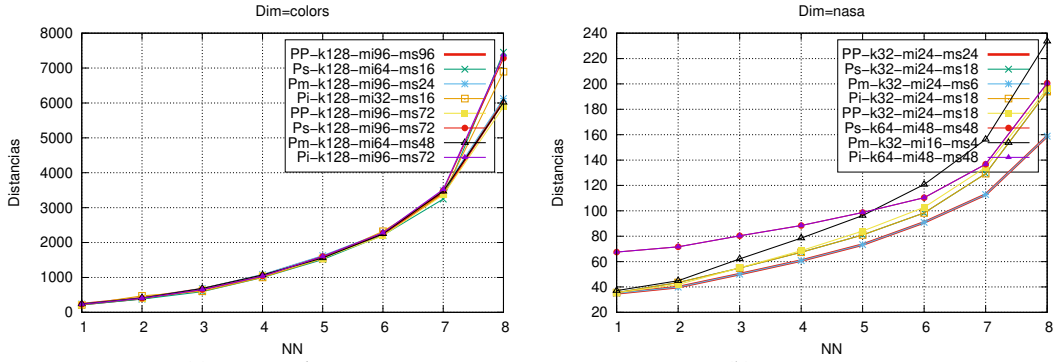
resultados por cada técnica.

La gráfica 5.34(a) corresponde a la base de datos *Colors*, nótese que en estas circunstancias la diferencia entre las técnicas empleadas es bastante pequeña, haciéndose más notoria al aumentar el número de vecinos a encontrar, en específico para 8. En 5.34(b) se encuentran los resultados elegidos para la base de datos *NASA*, véase que cada uno de los resultados de Ps requiere la misma cantidad de distancias que los correspondientes a Pi usando los mismos valores, mientras que para las técnicas Pm y PP sólo ocurre esto cuando los valores de k y m_i coinciden, sin embargo para Pm el valor m_s necesario es menor. Para los dos resultados restantes, se tiene mejor desempeño por parte de PP.

English

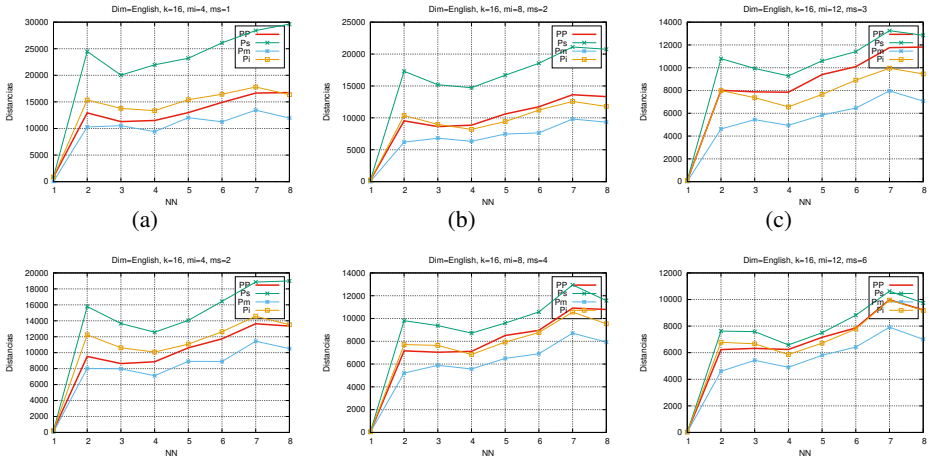
En la Figura 5.35 se presentan resultados de la base de datos *English* cuando $k = 16$, la primera columna contiene a $m_i = 4$, la segunda a $m_i = 8$ y la tercera a $m_i = 12$. De entre estas gráficas considérese 5.35(a) cuyos valores son $m_i = 4$ y $m_s = 1$, las distancias requeridas por Pm, Ps, PP y Pi para encontrar 8 vecinos son 11962, 29628, 16758 y 16365, respectivamente. De manera que Pm tiene mejor desempeño, también puede observarse en las otras gráficas. En 5.35(a) se muestra que es posible evitar calcular hasta un 26 % de las distancias necesarias para encontrar 8 vecinos, esto comparado con la técnica Pi.

En la Figura 5.36 se muestran resultados de la base de datos *English* cuando $k = 32$,



(a) $BD = colors$ (b) $BD = NASA$

Figura 5.34: Valores elegidos



(a) (b) (c)

(d) (e) (f)

Figura 5.35: BD=English, $k = 16$ y m_i={4,8,12}

$m_i = 8$ se localiza en la primera columna, $m_i = 16$ en la segunda y $m_i = 24$ en la tercera. Para ver el rendimiento considérese la gráfica 5.36(d) cuyos valores son $m_i = 8$ y $m_s = 4$, se tiene que para encontrar 1 vecino las distancias necesarias son 33 para Pm y 43 para Ps,PP y Pi, en el caso de 5 vecinos son 6601, 9688, 6616 y 6671. Pese a que es Pm la que tiene menor cantidad de distancias la diferencia con las otras técnicas ha disminuido considerablemente. En 5.36(d) se muestra que es posible evitar calcular hasta un 0.22 % de las distancias necesarias para encontrar 5 vecinos, esto comparado con la técnica PP.

La Figura 5.37 agrupa resultados de la base de datos *English* cuando $k = 64$, en la primera columna se tiene $m_i = 16$ en la segunda $m_i = 32$ y en la tercera $m_i = 48$. Considérese la gráfica 5.37(c) cuyos valores son $m_i = 48$ y $m_s = 12$, en este caso las distancias requeridas por Pm, Ps, PP y Pi para encontrar 7 vecinos son 2869, 3973, 4325, y 2836. Véase que para algunos valores la técnica que tiene mejor desempeño es Pi, para otros PP y en otros es Pm.

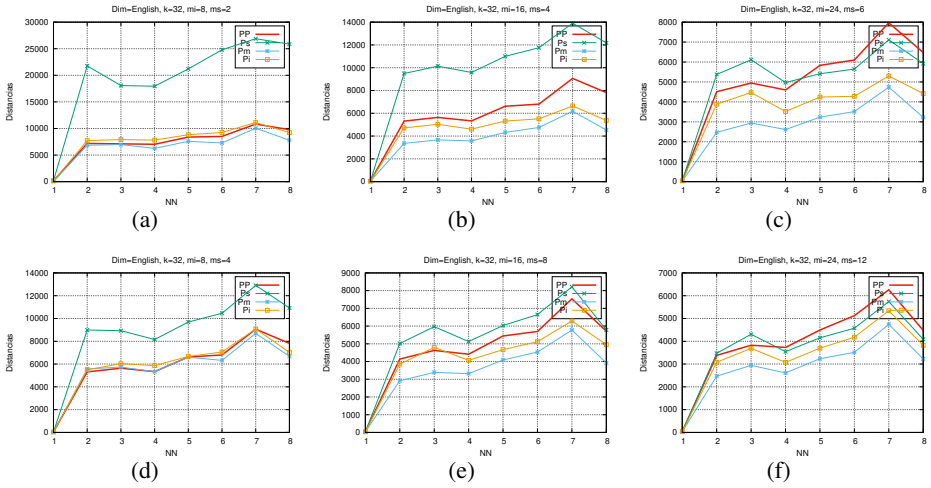


Figura 5.36: BD=English, $k = 32$ y m_i={8,16,24}

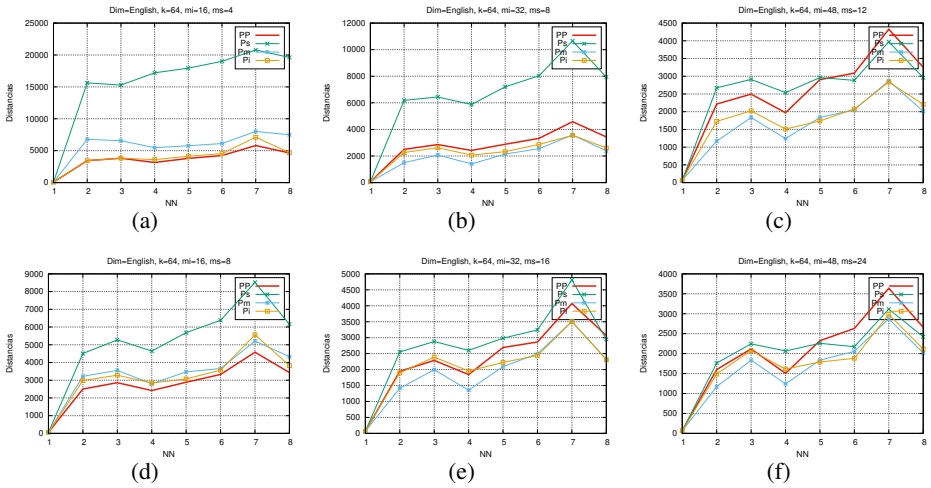


Figura 5.37: BD=English, $k = 64$ y m_i={16,32,48}

En la Figura 5.38 se muestran resultados de la base de datos *English* cuando $k = 128$, en la primera columna se tiene $m_i = 32$, en la segunda $m_i = 64$ y en la tercera 96. De entre estas gráficas véase 5.38(a) cuyos valores son $m_i = 32$ y $m_s = 8$, para la que las distancias requeridas por Pm, Ps, PP y Pi para encontrar 4 vecinos son 2848, 10938, 1538 y 1559. Nótese que en la mayoría de estos resultados es Pi la que tiene un mejor desempeño, aun cuando en algunos puntos es Pm y en otros lo es PP.

Spanish

En la Figura 5.39 se muestran resultados de la base de datos *Spanish* cuando $k = 16$, la primera columna contiene a $m_i = 4$, la segunda a $m_i = 8$ y la tercera $m_i = 12$. Para ver el rendimiento considérese la gráfica 5.39(a) cuyos valores son $m_i = 4$ y $m_s = 1$, las distancias que requieren Pm,

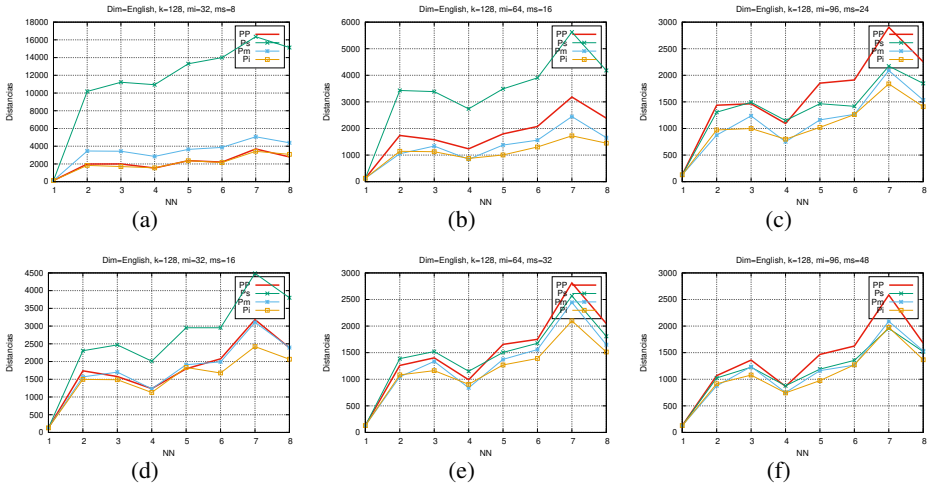


Figura 5.38: BD=English, k = 128 y m_i={32,64,96}

Ps, PP y Pi para encontrar 6 vecinos son 15947, 34872, 22655 y 23644, respectivamente. De manera que Pm es la técnica que se desempeña mejor. En 5.39(a) se muestra que es posible evitar calcular hasta un 29 % de las distancias necesarias para encontrar 6 vecinos, esto comparado con la técnica PP.

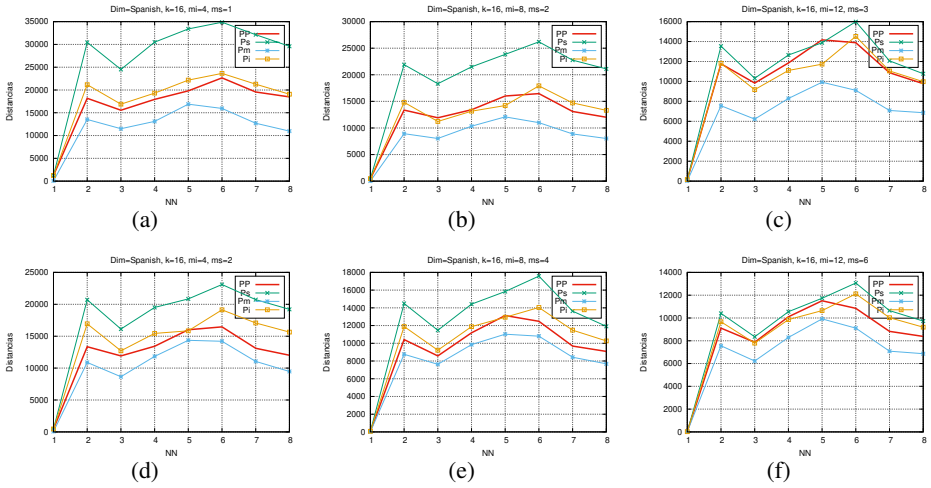


Figura 5.39: BD=Spanish, k = 16 y m_i={4,8,12}

La Figura 5.40 agrupa resultados de la base de datos *Spanish* cuando k = 32, en la primera columna se tiene m_i = 8, en la segunda m_i = 16 y en la tercera m_i = 24. De entre estas gráficas véase 5.40(a) cuyos valores son m_i = 8 y m_s = 2, las distancias que ocupan Pm, Ps, PP y Pi para encontrar 4 vecinos son 7875, 28764, 7926 y 9645. De manera que en este caso al inicio es Pm la de mejor desempeño y posteriormente es PP. En las otras gráficas es posible ver que es Pm la que tiene mejor desempeño, excepto en los últimos valores en 5.40(d), en los cuales es PP. En

5.40(a) se muestra que es posible evitar calcular hasta un 0.64 % de las distancias necesarias para encontrar 4 vecinos, esto comparado con la técnica PP.

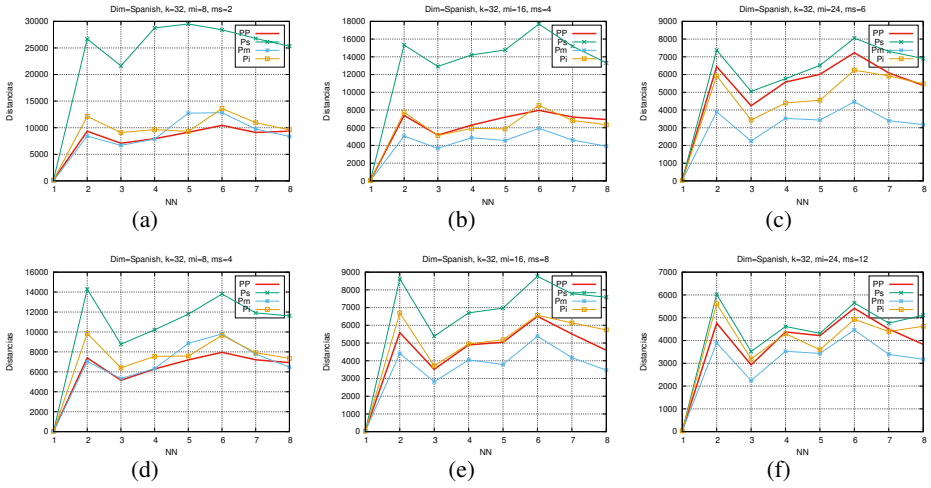


Figura 5.40: BD=Spanish, $k = 32$ y m_i={8,16,24}

En la Figura 5.41 se presentan resultados de la base de datos *Spanish* cuando $k = 64$, la primera columna contiene a $m_i = 16$, la segunda a $m_i = 32$ y la tercera a $m_i = 48$. Considérese la gráfica 5.41(b) cuyos valores son $m_i = 32$ y $m_s = 8$, para encontrar 3 vecinos las distancias requeridas por Pm, Ps, PP y Pi son 1332, 6558, 2335 y 2080. En este caso Pm se desempeña mejor, aunque puede verse que para los primeros valores de m_i es PP, e incluso para algunos puntos entre los últimos valores de m_i es Pi. En 5.41(b) se muestra que es posible evitar calcular hasta un 35 % de las distancias necesarias para encontrar 3 vecinos, esto comparado con la técnica Pi.

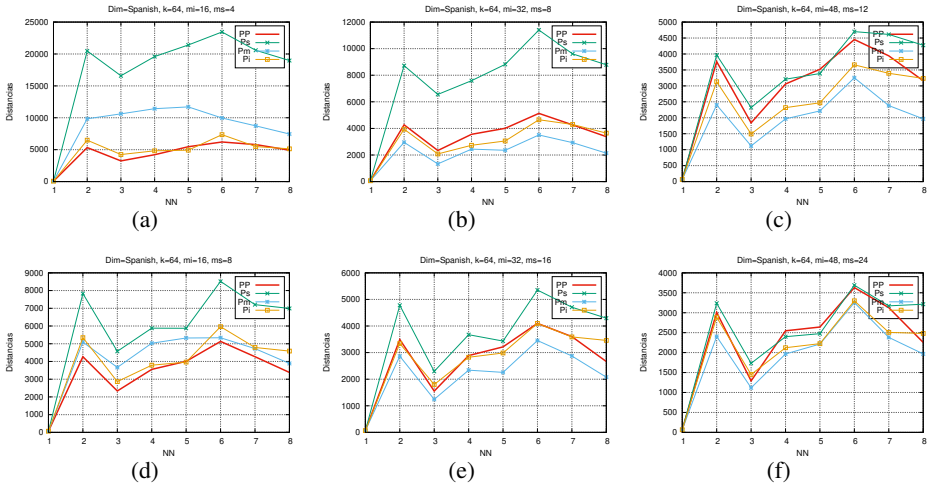
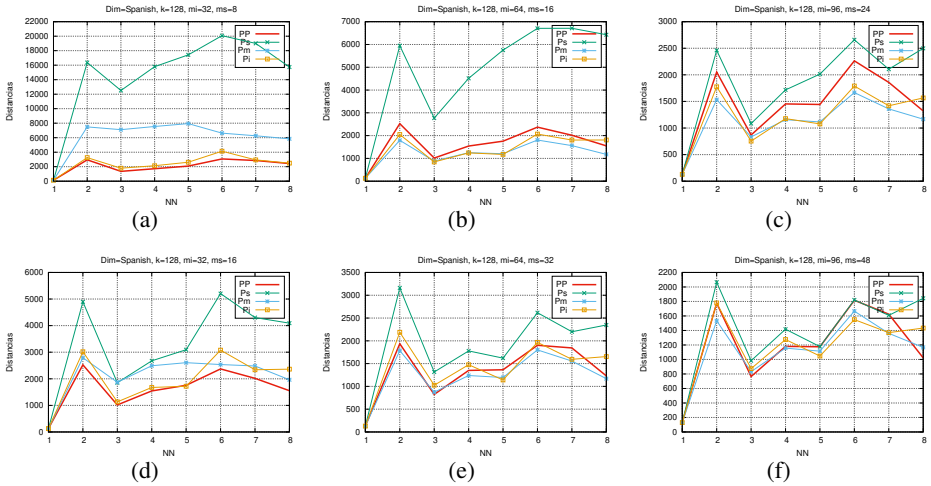


Figura 5.41: BD=Spanish, $k = 64$ y m_i={16,32,48}

En la Figura 5.42 se agrupan resultados de la base de datos *Spanish* cuando $k = 128$, la primera columna contiene a $m_i = 32$, la segunda a $m_i = 64$ y la tercera $m_i = 96$. De entre estas gráficas considérese 5.42(c) cuyos valores son $m_i = 96$ y $m_s = 24$, se tiene que las distancias necesarias para encontrar 4 vecinos son 1156, 1716, 1452 y 1176, correspondientes a Pm, PS, PP y Pi. En este caso se tiene que la técnica que se desempeña mejor es distinta en varios puntos, en algunos lo es Pm, en otros PP y en otros Pi.



(a) (b) (c)

(d) (e) (f)

Figura 5.42: BD=Spanish, $k = 128$ y m_i={32,64,96}

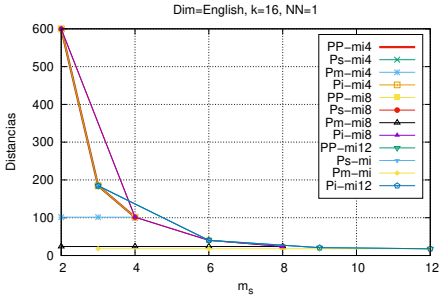
Variando m_s

En esta sección se presentan las Figuras 5.43 y 5.44, en las cuales se realiza la comparación de los resultados con respecto al parámetro m_s, en las bases de datos *English* y *Spanish* respectivamente. Se muestran los 4 permutantes en cada base de datos, además de los 3 valores de m_i por cada permutante, para encontrar 1 vecino más cercano.

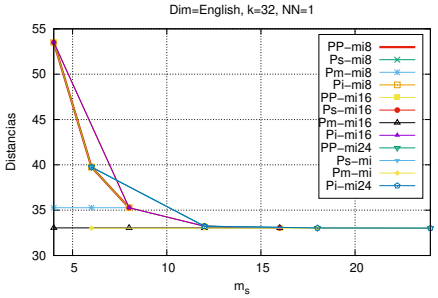
En la Figura 5.43 se agrupan las gráficas correspondientes a la base de datos *English*. El comportamiento es similar al de las bases de datos *Colors* y *NASA*, esto es, los resultados de Pm son constantes en cada m_i, mientras que PP, Ps y Pi ocupan la misma cantidad de distancias, y al aumentar los parámetros, k, m_i y m_s la diferencia entre los resultados disminuye considerablemente, incluso algunos al punto de ser iguales.

En la Figura 5.44 se agrupan las gráficas correspondientes a la base de datos *Spanish*. En estos resultados también se observa que Pm es constante en cada m_i, de igual forma PP, Ps y Pi ocupan la misma cantidad de distancias, y al aumentar los parámetros, k, m_i y m_s la diferencia entre los resultados disminuye considerablemente hasta tener el mismo número de distancias.

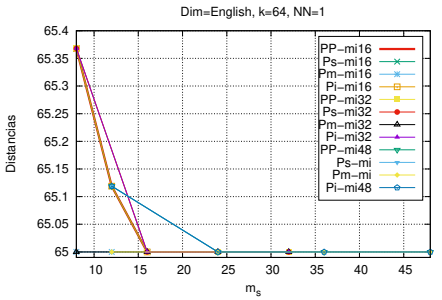
En la Figura 5.45 se agrupan los resultados elegidos para ambas bases de datos. Esto se generó a partir de lo que se examinó en las gráficas que se encuentran en 5.43 y 5.44. De modo



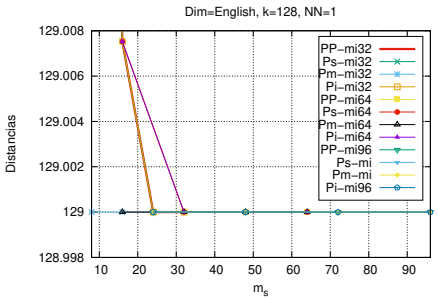
(a) $k = 16$, $m_i = \{4, 8, 12\}$



(b) $k = 32$, $m_i = \{8, 16, 24\}$

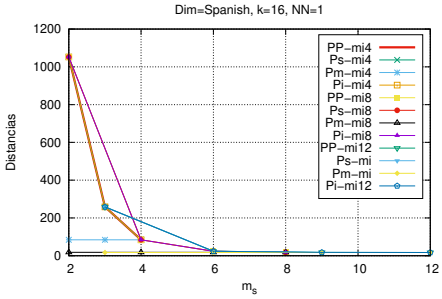


(c) $k = 64$, $m_i = \{16, 32, 48\}$

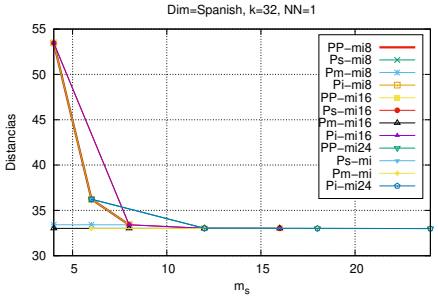


(d) $k = 128$, $m_i = \{32, 64, 96\}$

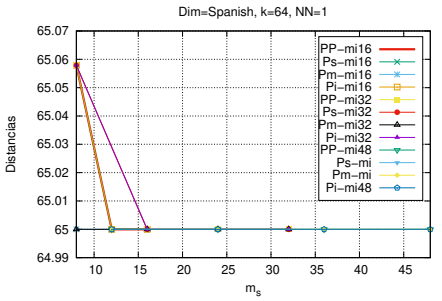
Figura 5.43: Variando m_s en BD=English



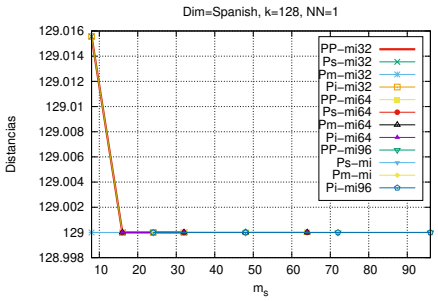
(a) $k = 16$, $m_i = \{4, 8, 12\}$



(b) $k = 32$, $m_i = \{8, 16, 24\}$



(c) $k = 64$, $m_i = \{16, 32, 48\}$

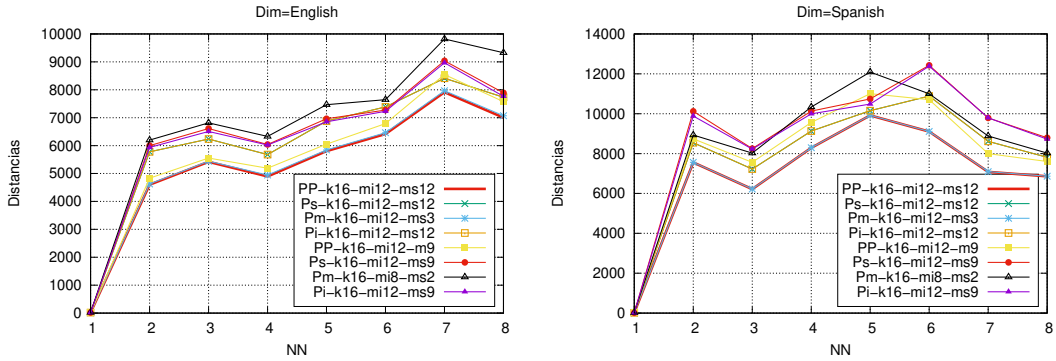


(d) $k = 128$, $m_i = \{32, 64, 96\}$

Figura 5.44: Variando m_s en BD=Spanish

que se comparan las técnicas usando los parámetros que generan la menor cantidad de distancias. Se eligieron dos resultados por cada técnica.

En 5.45(a) corresponde a la base de datos *English*, nótese que en estas circunstancias uno de los resultados de la técnica Pm ($k = 16$, $m_i = 12$ y $m_s = 3$) requiere la misma cantidad de distancias que PP hasta encontrar 4 vecinos después se ve un ligero aumento para Pm, en cambio el otro resultado ($k = 16$, $m_i = 8$ y $m_s = 2$) tiene el mayor número de distancias ocupadas. En 5.45(b) se encuentran los resultados elegidos para la base de datos *Spanish*, en este caso se tiene que uno de los resultados ($k = 16$, $m_i = 12$ y $m_s = 3$) ocupa la misma cantidad de distancias que PP, para el segundo resultado ($k = 16$, $m_i = 8$ y $m_s = 2$) en algunos punto es mayor y en otros menor que Ps y Pi. En ambas bases de datos, Pm la cantidad de comparaciones que necesita es similar a las requeridas por PP, no obstante el valor de m_s que requieren es menor, $m_s = 3$ en ambas bases.



(a) *BD = English* (b) *BD = S panish*

Figura 5.45: Valores elegidos

Parte V

CONCLUSIONES

Capítulo 6

Conclusiones

En las bases de datos multimedia, para realizar las búsquedas por similitud implica encontrar los elementos que son más parecidos a una consulta dada, es necesaria una función de distancia que determina los elementos más parecidos entre sí, dicha función es definida de acuerdo al tipo de la base de datos, ésto con la ayuda de un experto en ese campo. Es posible modelar este tipo de problemas como un espacio métrico con la finalidad de agilizar la búsqueda y diminuir su costo.

Entre los algoritmos propuestos para la búsqueda por similitud se encuentran los algoritmos basados en permutaciones, en los cuales la idea principal es seleccionar un conjunto de elementos definidos como *permutantes*, para posteriormente calcular la distancia de cada uno de los objetos de la base de datos a cada uno de los permutantes y así generar su *permutación*, ordenándolos de forma creciente.

En este trabajo se realizó una continuación de lo expuesto en [FRCI20]. De tal forma que esta tesis se enfocó en emplear dos técnicas usando índices invertidos:

- La primera propuesta utiliza un nuevo índice, donde las listas contienen sólo aquellos objetos que tienen un permutante entre sus primeros m_i más cercanos, la lista de candidatos es la unión de las listas de cada permutante p de $\Pi_q^{-1}(p) \leq m_s$. Además se presenta una nueva distancia entre permutantes.
- En la segunda propuesta se emplea el índice invertido tal como en [AS10], modificando sólo el cálculo de la distancia entre permutantes.

De entre las dos propuestas la primera experimenta un mejor desempeño, incluso comparada con ciertos elementos de los resultados obtenidos por la técnica de *permutantes*. La primera propuesta funciona mejor cuando los parámetros usados son pequeños, se observó experimentalmente que en las bases de datos sintéticas k debe ser menor o igual que el doble del valor de la

dimensión usada y en las bases de datos reales también se obtiene mejor resultado cuando k es pequeño. Para la mayoría de los resultados que tienen mejor desempeño los parámetros se ven de la forma siguiente: $\frac{k}{4} \leq m_i \leq \frac{3k}{4}$ y $m_s \leq \frac{m_i}{2}$, mientras que para unos cuantos se ven de la siguiente forma: $m_i \leq \frac{3k}{4}$ y $m_s \leq \frac{3m_i}{4}$, estos valores pueden percibirse en ambos tipos de bases de datos. También se observó, experimentalmente, que es posible evitar calcular hasta un 67 % de las distancias para algunos de los casos presentados.

Para la segunda propuesta se observó que se desempeña mejor a medida que m_i y m_s aumentan, siendo más notorio en los resultados para las bases de datos *Colors* y *NASA*, en especial en ésta última y cuando $m_s = m_i$.

Bibliografía

[AS10] Giuseppe Amato and Pasquale Savino. Approximate similarity search in metric spaces using inverted files. ICST, 5 2010.

[BBK+01a] S. Berchtold, C. Böhm, D. Keim, F. Krebs, and H.-P. Kriegel. On optimizing nearest neighbor queries in high-dimensional data spaces. In *8th International Conference Database Theory (ICDT)*, volume 1973, pages 435–449. Springer, 2001.

[BBK01b] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[Ben75] J. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

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

[BKK96] S. Berchtold, D. Keim, and H. Kriegel. The X-tree: an index structure for high-dimensional data. In *Proceedings 22nd Conference on Very Large Databases (VLDB'96)*, pages 28–39, 1996.

[CFN05] Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In Alexander Gelbukh, Álvaro de Albornoz, and Hugo Terashima-Marín, editors, *MICAI 2005: Advances in Artificial Intelligence*, pages 405–414, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[CNBYM99] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. Technical Report TR/DCC-99-3, Dept. of Computer Science, Univ. of Chile, 1999. `ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/-survmetric.ps.gz`.

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

[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.

[FKS02] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17, 10 2002.

[FRCI20] Karina Figueroa, Nora Reyes, and Antonio Camarena-Ibarrola. Candidate list obtained from metric inverted index for similarity searching. In Lourdes Martínez-Villaseñor, Oscar Herrera-Alcántara, Hiram Ponce, and Félix A. Castro-Espinoza, editors, *Advances in Computational Intelligence*, pages 29–38, Cham, 2020. Springer International Publishing.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

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

[HS03] Gisli R Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces (survey article). *ACM Transactions on Database Systems (TODS)*, 28:517–580, December 2003. ACM ID: 958948.

[Rud76] W. Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976.