UNIVERSIDADE DA CORUÑA

Departamento de Computación

MÉTODOS DE MEJORA DEL RENDIMIENTO EN BÚSQUEDAS POR SIMILITUD SOBRE ESPACIOS MÉTRICOS

Tese Doutoral

Doutorando:

Luis Andrés González Ares

Directores:

Óscar Pedreira Fernández, Nieves Rodríguez Brisaboa

A Coruña, xullo de 2012

Ph. D. Thesis supervised by
Tese doutoral dirixida por

Óscar Pedreira Fernández
Departamento de Computación
Facultade de Informática
Universidade da Coruña
15071 A Coruña (España)
Tel: +34 981 167000 ext. 1306
Fax: +34 981 167160
opedreira@udc.es

Nieves Rodríguez Brisaboa
Departamento de Computación
Facultade de Informática
Universidade da Coruña
15071 A Coruña (España)
Tel: +34 981 167000 ext. 1243
Fax: +34 981 167160
brisaboa@udc.es

A mi familia

Agradecimientos

A las personas:

Primero me parece necesario reconocer a los directores de este trabajo, ya que su guía fue determinante en el mismo. A Nieves por ofrecerme la oportunidad de investigar en un tema que me ha resultado atractivo y motivador, y por animarme continuamente, y a Óscar por su apoyo y disponibilidad constante, por su paciencia mientras me incorporaba a una temática que él lideraba y porque las discusiones con él siempre fueron fuente de inspiración y de motivación.

A los compañeros y amigos del Laboratorio de Bases de Datos y de Enxenio, y a personas de otros grupos ligadas de diferentes formas al nuestro, les debo agradecimiento por hacerme compartir buenos momentos a lo largo del tiempo y por el aprecio que siempre me han mostrado. Aunque la lista sería larga, pues hay relaciones de amistad y compañerismo que en algunos casos abarcan décadas, mientras que otras irrumpieron con intensidad desde hace poco tiempo, solo voy a destacar a quienes de alguna forma han contribuido a este trabajo, aunque puede que en algún caso no fueran conscientes de ello. A Sebas porque si no se hubiera encargado de unas horas de clase este trabajo se entregaría inevitablemente más tarde, a Susana por sus ayudas diversas y por estar pendiente de todos, a Alberto por ofrecerse cuando ni tiempo tenía, y a Antonio y a Luisa por muchas más cosas de las que ellos piensan.

A la comunidad investigadora en Búsqueda por Similitud en Espacios Métricos y especialmente a la "escuela chilena", que abarca a personas de varios países, ya que el análisis de sus trabajos me ha resultado agradable y enriquecedor.

Quiero agradecer especialmente a mi familia su constante apoyo y comprensión en el desarrollo de este trabajo, soportando mi dedicación al mismo, que a veces fue muy intensa.

A las herramientas utilizadas y a quienes las potencian:

De mi larga etapa profesional al margen de la docencia, además de muchas otras cosas que aprendí gracias a los lugares por los que he pasado y a las personas con quienes trabajé, me queda también un bagaje en toda una serie de herramientas ligadas a Unix y hoy consideradas software libre, que junto a otras más recientes han contribuido fundamentalmente a este trabajo, porque en él solo se ha utilizado este tipo de software por razones de calidad en los resultados. A eso habría que añadir mi reconocimiento especial a toda la comunidad que hace posible que podamos trabajar con una herramienta como Latex.

A la música:

La música me ha ayudado en las muchas y largas horas que un trabajo así requiere. A veces lo más novedoso, lo más conocido, y otras mis gustos más personales y rebuscados, como el jazz, el rock progresivo y la música celta, han sido una gran ayuda en el proceso de hilvanar ideas y palabras.

Resumen

En esta tesis se abordan problemas de rendimiento de las búsquedas por similitud en espacios métricos.

La búsqueda por similitud tiene como finalidad determinar los objetos más semejantes o cercanos a uno dado. Los espacios métricos permiten formalizar dicha búsqueda y han dado lugar a métodos, cuyo objetivo principal es reducir el número de evaluaciones de la función de distancia, intentando descartar el mayor número posible de objetos o de zonas que representan. Las soluciones existentes son métodos basados en pivotes, que obtienen un número reducido de evaluaciones pero requieren cantidades importantes de espacio, y métodos basados en particiones, que necesitan poco espacio pero que incrementan el número de evaluaciones.

Las contribuciones de esta tesis son: i) un nuevo método basado en pivotes que reduce el tamaño del índice gracias a que almacena, para cada objeto, la distancia al pivote más prometedor para descartarlo, manteniendo un número de evaluaciones de la función de distancia que lo hacen competitivo con los métodos de particiones; y ii) una nueva estrategia para métodos basados en particiones que, reduciendo progresivamente el tamaño del cluster, disminuye significativamente el número de evaluaciones de la función de distancia, al explorar los clusters que no han sido descartados.

Resumo

Nesta tese abórdanse problemas de rendemento das procuras por similitude en espazos métricos.

A procura por similitude ten como finalidade determinar os obxectos máis semellantes ou próximos a un dado. Os espazos métricos permiten formalizar dita procura e deron lugar a métodos, cuxo obxectivo principal é reducir o número de avaliacións da función de distancia, tentando descartar o maior número posible de obxectos ou de zonas que representan. As solucións existentes son métodos baseados en pivotes, que obteñen un número reducido de avaliacións pero requiren cantidades importantes de espazo, e métodos baseados en particións, que necesitan pouco espazo pero que incrementan o número de avaliacións.

As contribucións desta tese son: i) un novo método baseado en pivotes que reduce o tamaño do índice grazas a que almacena, para cada obxecto, a distancia ao pivote máis prometedor para descartalo, mantendo un número de avaliacións da función de distancia que o fan competitivo cos métodos de particións; e ii) unha nova estratexia para métodos baseados en particións que, reducindo progresivamente o tamaño do cluster, diminúe moito o número de avaliacións da función de distancia, ao explorar os clusters que non foron descartados.

Abstract

In this thesis performance problems of similarity search in metric spaces are considered.

The aim of the similarity search is to determine the most similar or closer objects to one given. The metric spaces allow to formalize this search and they have given rise to methods, whose main objective is to reduce the number of evaluations of the distance function, trying to discard the greater possible number of objects or of zones that they represent. The existing solutions are pivot-based methods, that obtain a reduced number of evaluations but require significant amounts of space; and clustering-based methods, that need little space but increase the number of evaluations.

The contributions of this thesis are: i) a new pivot-based method that reduces the size of the index because it stores, for every object, the distance to the most promising pivot to discard it, maintaining a number of evaluations of the function of distance that make it competitive with clustering-based methods; and ii) a new strategy for clustering-based methods that, reducing progressively the size of the cluster, diminishes significantly the number of evaluations of the distance function when it explores the clusters that have not been discarded.

Índice general

Índice de figuras

Índice de tablas

Índice de lemas

Capítulo 1

Introducción

Los sistemas de información necesitan realizar de manera eficiente la búsqueda de los datos a través de las estructuras de almacenamiento. Dicha búsqueda consiste en obtener los elementos que cumplen una serie de condiciones, lo que se conoce como el *criterio de búsqueda*.

En el caso más habitual se tiene la denominada *búsqueda exacta*, en la que el criterio de búsqueda es una condición de igualdad o de desigualdad, que está completamente determinada mediante operadores algebraicos. Este tipo de búsqueda es válido para resolver los problemas planteados en muchas aplicaciones de diversos tipos y temáticas, pero resulta inadecuado para aplicar en casos en los que el criterio de búsqueda, basado en la exactitud, se transforma en otro determinado por características como la similitud o la cercanía. Se tiene así otro tipo de búsqueda en la que se determinan los objetos más cercanos o más próximos a un elemento, denominada *busqueda por similitud* o *búsqueda por proximidad*. Ejemplos de esta búsqueda son frecuentes en las aplicaciones en las que los datos no poseen la rígida estructuración que conllevan las bases de datos tradicionales.

Los espacios métricos constituyen un modelo matemático que permite formalizar la búsqueda por similitud gracias a la definición de una función de distancia, proporcionando así una base sólida sobre la que plantear métodos de búsqueda que sean eficientes.

En este capítulo se contextualiza la búsqueda por similitud sobre espacios métricos, se describen problemas que inciden en su rendimiento, se plantean objetivos para solucionar algunos de sus problemas y se describen las contribuciones de esta tesis en la solución de dichos problemas.

1.1. Motivación

El desarrollo del tratamiento de la información ha abarcado durante las últimas décadas todo tipo de disciplinas y actividades humanas, originando un volumen de datos en constante crecimiento y tipos de datos complejos que incluyen documentos en diversos formatos y de variada extensión, y vínculos entre ellos. Ámbitos como los sistemas multimedia, la biología molecular, el diseño asistido por ordenador, las bibliotecas digitales y los sistemas de seguimiento y de recomendación, como los de las actividades industriales, financieras, sanitarias y sociales, ofrecen múltiples ejemplos de dominios en los que se generan, almacenan y procesan datos denominados *semiestructurados* y *no estructurados*, donde los criterios de búsqueda no se basan en la exactitud y sí en la similitud.

Los mecanismos que se emplean en la búsqueda por similitud difieren de los utilizados en la búsqueda exacta, ya que a la última se le pueden aplicar operaciones completamente determinadas, por ejemplo igual a, mayor que, etc. La utilización de un criterio concreto permite determinar con facilidad si un elemento lo cumple o no, simplificando los algoritmos de búsqueda a utilizar. Pero frente a la concreción de los criterios utilizados en la búsqueda exacta, la búsqueda por similitud requiere, en primer lugar, un mecanismo para determinar con rigurosidad el concepto de similitud, que evite problemas de inconcreciones o de subjetividad, y posteriormente, la definición de algoritmos que apliquen ese mecanismo y que, por lo general, son diferentes a los utilizados en la búsqueda exacta.

Algunas soluciones adoptadas utilizan características dependientes del dominio de aplicación, que suelen ser poco extensibles a otros ámbitos, como ocurre, por ejemplo, en las soluciones de búsqueda por proximidad en bases de datos espaciales. En otros casos se utilizan fundamentos más genéricos y versátiles, como por ejemplo, los espacios métricos, que asocian una función de distancia a los elementos de un conjunto ($d : X \times X \longrightarrow \mathbb{R}$), cumpliendo las propiedades de identidad ($\forall\ x, y \in X,\ d(x, y) = 0 \ \Leftrightarrow\ x = y$), simetría ($\forall\ x, y \in X,\ d(x, y) = d(y, x)$) y la que es de mayor aplicación en los métodos de búsqueda, la propiedad de la desigualdad triangular ($\forall\ x, y, z \in X,\ d(x, y) \leq d(x, z) + d(z, y)$).

Los espacios métricos aportan un marco teórico que permite implementar la búsqueda por similitud con rigurosidad y hacerlo además de manera genérica, siendo aplicable a cualquier conjunto de datos sobre el que se defina una función que cumpla esas propiedades. Se pueden utilizar además varias funciones de distancia para un mismo conjunto, lo que amplía las posibilidades, ya que permite calibrar mejor qué función se ajusta con más exactitud a las características de los datos y al concepto de similitud que hay en ellos.

En unos casos las búsquedas se realizan sobre colecciones que no sufren modificaciones, mientras que en otros casos se requiere una capacidad de dinamismo que permita modificar el conjunto inicial, adaptándose a veces a los resultados obtenidos a medida que se realizan las operaciones de búsqueda.

La manera trivial de resolver una búsqueda por similitud consiste en calcular la distancia existente entre el objeto de búsqueda y cada elemento de la base de datos. Las características de los tipos de datos utilizados, a menudo complejas, como es el caso de imágenes o de cadenas de ADN, provocan que la comparación de dos objetos mediante la función de distancia tenga, en general, un coste considerable, por lo que la comparación directa de la distancia entre el objeto de búsqueda y los objetos de la base de datos, conlleva un proceso demasiado costoso que lo convierten en inviable para su aplicación práctica.

Siempre que se plantea el problema del acceso a los datos, se considera la posibilidad de crear estructuras auxiliares denominadas *índices*, con la finalidad de acelerar las operaciones de búsqueda. Normalmente estos índices se crean en un momento previo al de realización de las consultas, por lo que su creación no se considera una parte del proceso de búsqueda y el coste de la creación del índice no se vincula al coste de la búsqueda.

En las búsquedas por similitud sobre espacios métricos se realiza un proceso previo de los datos para crear un índice en el que se almacenan, fundamentalmente, las distancias entre los objetos de la base de datos y algunos objetos de referencia, con el objetivo de utilizar esas distancias en el proceso de búsqueda, intentando resolver las consultas sin comparar el objeto de búsqueda con cada elemento de la base de datos, aplicando para ello las propiedades de los espacios métricos, especialmente la desigualdad triangular.

Una consulta por similitud se expresa mediante un objeto de consulta $q \in X$ y un tipo de búsqueda sobre ese objeto. Los tipos de búsquedas que se presentan con más frecuencia son la *búsqueda por rango*, $R(q, r)$, que determina los objetos que se encuentran a una distancia de q menor o igual que r, y la *búsqueda de los k-vecinos más cercanos*, $kNN(q)$, que obtiene los k elementos más cercanos a q.

La aplicación de los espacios métricos como marco teórico para la resolución de la búsqueda por similitud, ha dado lugar a un elevado número de métodos, que presentan diferencias notables en sus procedimientos. Los métodos pueden clasificarse en dos grupos, los basados en pivotes y los basados en particiones o de clustering [Chávez y otros, 2001b].

Los *métodos basados en pivotes* seleccionan un subconjunto de objetos de la colección, a los que se denomina *pivotes*, y almacenan en el índice las distancias de los pivotes al resto de objetos. Al realizar una búsqueda, comparan el objeto de búsqueda q con los pivotes, y utilizan las distancias almacenadas en el índice para

descartar el mayor número posible de objetos sin compararlos con q. Si un objeto no se descarta con un pivote, se prueba con otro, y si finalmente no logra descartarse con ningún pivote, hay que calcular su distancia a q para comprobar si realmente cumple el criterio de búsqueda.

La complejidad de la búsqueda viene dada por la suma de la complejidad interna, que en el caso de usar pivotes es el número de evaluaciones de la función de distancia entre el objeto de búsqueda y los pivotes, y de la complejidad externa, que es el número de comparaciones del objeto de búsqueda con los objetos que no resultan descartados.

Los *métodos basados en particiones* dividen el espacio en conjuntos disjuntos denominados *clusters*, definidos en función de unos objetos de referencia denominados *centros* y de, o bien sus distancias a los objetos de otros clusters, si usan el criterio del hiperplano, o bien de la distancia al objeto del cluster más alejado de su centro, llamada *radio de cobertura*. En el índice se almacenan los datos relevantes sobre cada cluster, como el centro y las distancias que determinan los límites del cluster. Cada objeto de la base de datos pertenece al cluster que corresponde a su centro más cercano. Dada una consulta, se compara el objeto de búsqueda q con los centros de cada cluster, y se puede descartar un cluster, sin comparar sus objetos con q, si los datos del índice garantizan que ninguno de sus elementos cumple el criterio de búsqueda. Si un cluster no puede descartarse, se calcula la distancia de cada uno de sus elementos a q para comprobar si cumple el criterio de búsqueda, en un proceso de exploración exhaustiva.

La complejidad interna es ahora el número de evaluaciones de la función de distancia necesarias para comparar el objeto de búsqueda con los centros de los clusters, mientras que la complejidad externa es el número de evaluaciones de la función de distancia que se realizan en el proceso de exploración de los clusters no descartados.

Los métodos basados en pivotes obtienen valores reducidos en el número de evaluaciones de la función de distancia, pero a costa de necesitar grandes cantidades de espacio en el tamaño del índice, debido al número de distancias precalculadas que almacena. Los métodos basados en particiones tienen un comportamiento completamente diferente, ya que generan índices de pequeño tamaño, dado que la información que almacenan se reduce a unas características mínimas de cada cluster, pero por contra, requieren un número de evaluaciones de la función de distancia mayor que los métodos de pivotes.

En general, los métodos tratan de reducir el número de evaluaciones de la función de distancia, ya que se considera que es el factor más importante en el coste total de las búsquedas, por lo que a menudo se utiliza ese valor como el único criterio para determinar la eficiencia de un método. Sin embargo, hay otros factores involucrados en el coste total, como son el tiempo de entrada/salida y el tiempo de CPU necesario

para procesar los datos almacenados en el índice, que son importantes de cara al rendimiento global de los métodos, hasta el punto de que pueden llegar a hacer inviable su aplicación práctica en casos reales, sobre todo si el volumen de datos a tratar es elevado.

A continuación se describen algunos problemas que tienen una incidencia importante en el rendimiento de los métodos.

1.1.1. Problemas que inciden en el rendimiento

La selección de los objetos de referencia

La selección de los objetos de referencia, esto es, de los pivotes en los métodos basados en pivotes, y de los centros de los clusters en los métodos de clustering, tiene una gran influencia en el rendimiento obtenido, como han determinado varios estudios que se referencian en la sección 4.1.2.

Muchos métodos optan por una elección aleatoria, pero este criterio de selección puede llevar a situaciones que ocasionan problemas de rendimiento, como que el conjunto de objetos elegidos pueda presentar un sesgo con respecto al resto de los objetos de la colección, o que se elijan objetos muy cercanos entre sí.

Se han propuesto varias alternativas para la elección de los objetos de referencia, aunque casi todas ellas referidas a pivotes, y algunas de ellas resuelven el problema de manera eficiente. Un nuevo método debe tener este hecho en cuenta.

Además del criterio de selección, otro punto importante es el número de objetos seleccionados, que tiene una incidencia directa en el rendimiento porque determina la complejidad interna, ya que, cuantos más objetos se seleccionen, más comparaciones con el objeto de búsqueda se realizarán.

Los métodos de búsqueda por similitud sobre espacios métricos se basan en determinar un conjunto de objetos de referencia que sean eficaces en el proceso de descarte, pero en todo caso, tiene que tratarse de un número reducido de objetos, porque si no fuera así, la complejidad interna se elevaría, acercándose a la que se obtendría resolviendo de manera trivial la búsqueda, esto es, comparando directamente el objeto de búsqueda con cada objeto de la base de datos, lo que haría inútil el planteamiento de los métodos.

El tamaño del índice en los métodos basados en pivotes

Los métodos basados en pivotes presentan la desventaja de que generan índices que ocupan gran cantidad de espacio, debido a que almacenan un número alto de

distancias entre los objetos de la base de datos, pero ese hecho es lo que les permite ser eficientes en el proceso de descarte de los objetos, y reducir con ello el número de evaluaciones de la función de distancia.

El espacio ocupado por el índice, además de estar relacionado con el tamaño de la memoria principal, afecta al coste computacional que se origina al procesar sus datos, pudiendo llegar a ocurrir que dicho coste sea incluso mayor que el producido por una búsqueda secuencial. Esta circunstancia puede ocurrir al aplicar métodos basados en pivotes sobre colecciones de gran tamaño. Otra circunstancia que se observa experimentalmente es que los métodos basados en pivotes no tienen, en general, un buen comportamiento al tratar grandes dimensiones. Todo ello puede hacer que métodos considerados habitualmente como muy eficientes, realmente no lo sean al aplicarlos a determinados casos reales.

La exploración exhaustiva de los clusters no descartados

Los métodos basados en particiones tienen la peculiaridad de que pueden descartar la totalidad de un cluster sin comparar el objeto de búsqueda con los elementos que forman parte del mismo. Esta es una cualidad importante, ya que permite descartar muchos objetos a cambio de almacenar una información muy reducida en el índice y necesitar comparar únicamente el objeto de búsqueda con el centro del cluster.

Pero si el cluster no resulta descartado, no queda más remedio que calcular la distancia existente entre cada uno de sus elementos y el objeto de búsqueda, en una exploración exhaustiva del cluster, lo que origina un número considerable de evaluaciones de la función de distancia, incrementando la complejidad externa.

Los métodos basados en clustering actúan, en general, de esta forma, ya que no disponen de más información sobre los objetos, que el cluster al que pertenecen.

Hay que resaltar el hecho de que habitualmente una buena parte de estas comparaciones resulta infructuosa, ya que se trata de determinar si los objetos que pertenecen a un cluster cumplen un criterio de búsqueda, pero lo más probable es que solo una pequeña parte de esos objetos lo cumpla mientras que el resto de los objetos no lo hará. Sin embargo, estos objetos que no cumplen el criterio elevarán notoriamente el número de comparaciones de la función de distancia.

En este punto incide también el número de objetos que tiene el cluster, denominado frecuentemente *tamaño del cluster*. Cuanto mayor sea este número, por tanto cuantos más objetos tengan los clusters, más posibilidades habrá de que se genere un número mayor de comparaciones infructuosas al explorar los clusters no descartados.

1.2. Objetivos

Después de describir algunos de los problemas que presentan los métodos de búsqueda por similitud sobre espacios métricos, en esta sección se exponen los objetivos planteados en esta tesis:

1. Desarrollar un método de búsqueda por similitud basado en pivotes que logre una importante reducción del tamaño del índice y que a la vez origine un número aceptable de evaluaciones de la función de distancia, hasta el punto de poder compararse competitivamente, en ambos indicadores, con los métodos basados en particiones.

   Se utilizará además un criterio de selección de pivotes que sea efectivo para el descarte de cada objeto de la base de datos.

2. Desarrollar una estrategia de búsqueda por similitud, aplicable a los métodos basados en particiones, que reduzca el número de evaluaciones de la función de distancia al explorar los clusters que no resultan descartados, esto es, que reduzca la complejidad externa.

   Aunque es previsible que el tamaño del índice se incremente, debe mantenerse en unos valores cercanos a los ya existentes, y en todo caso, que resulten aceptables para su aplicación a casos reales.

Estos objetivos se centran en buscar soluciones a los puntos débiles de ambos tipos de métodos, al tamaño del índice en el caso de los métodos basados en pivotes, y al número de evaluaciones de la función de distancia en el caso de los métodos basados en particiones.

1.3. Contribuciones

Las contribuciones de esta tesis se centran en aportar soluciones a los objetivos planteados. A continuación se realiza una breve descripción de dichas contribuciones, mencionando sus principales resultados.

Índice de pivote único (UPI)

La primera contribución es un nuevo método de búsqueda por similitud basado en pivotes, denominado *Índice de Pivote Único (UPI)*, que reduce de manera muy significativa el tamaño del índice.

El nuevo método se ha basado en criterios formales y en análisis experimentales para determinar la eficacia que tienen los pivotes en el proceso de descarte de cada uno de los objetos de la base de datos. Se ha aplicado un método de selección de pivotes que garantiza que se obtienen pivotes cercanos y lejanos a cada objeto, y se han caracterizado las colecciones utilizadas, estableciendo los valores que determinan que los pivotes más cercanos y más lejanos a un objeto son efectivos para descartarlo. El nuevo método consiste en almacenar en el índice, para cada objeto de la base de datos, únicamente la distancia a un pivote, que es el más cercano o el más lejano al objeto, según que sus valores se correspondan con los resultantes en la caracterización de la colección efectuada previamente.

Los resultados experimentales muestran que, en la reducción del espacio del índice, UPI mejora de manera importante a otros métodos basados en pivotes, y que compite, casi siempre favorablemente, con los métodos de clustering, tanto en el tamaño del índice como en el número de evaluaciones de la función de distancia.

Gracias a sus cualidades, el nuevo método permite abordar el tratamiento real de colecciones extensas que algunos otros métodos basados en pivotes no podían afrontar, además de ser una alternativa viable a los métodos basados en particiones, por ejemplo, para el tratamiento de grandes dimensiones.

Reducción de Cluster (CR)

La segunda aportación es una nueva estrategia aplicable a los métodos de búsqueda por similitud basados en particiones, denominada *Reducción de Cluster (CR)*, que evita la exploración exhaustiva de los clusters no descartados, reduciendo así el número de evaluaciones de la función de distancia.

Consiste en definir un número de regiones dentro de cada cluster, de forma que, ante una búsqueda en la que un cluster no resulta descartado, en vez de calcular la distancia del objeto de búsqueda a cada objeto del cluster, se restringe la exploración a las regiones del cluster, que se van descartando a medida que se comparan sus objetos con el objeto de búsqueda. El cluster se explora así región por región, llegando un momento en el que el resto de sus regiones pueden descartarse sin comparar sus objetos con el objeto de búsqueda.

La propuesta incluye los algoritmos de búsqueda por rango y de búsqueda kNN, y su aplicación origina una reducción del número de evaluaciones de la función de distancia, que es un punto débil de los métodos basados en particiones.

La estrategia puede aplicarse a cualquier método basado en particiones que use el radio de cobertura como criterio de descarte. Esta estrategia posee además características dinámicas.

1.4. Estructura de la tesis

Esta tesis presenta la siguiente estructura:

- En el capítulo 2 se realiza una introducción a los conceptos fundamentales de espacios métricos, y especialmente a los de la búsqueda por similitud, describiendo las estrategias utilizadas en ella e incluyendo los fundamentos formales que se aplican en los capítulos siguientes.
- En el capítulo 3 se estudian algunos de los métodos más relevantes de búsqueda por similitud sobre espacios métricos, tanto basados en pivotes como en particiones.
- El capítulo 4 presenta el método Índice de Pivote Único (UPI), partiendo de la exposición de los problemas de los métodos basados en pivotes, mencionando después las soluciones existentes, analizando las características que deben cumplir los pivotes para ser eficaces en el proceso de descarte, para a continuación exponer el método, describir sus algoritmos de búsqueda por rango y kNN, evaluarlo experimentalmente y analizar sus resultados.
- El capítulo 5 presenta la estrategia Reducción de Cluster (CR), planteando primero los problemas existentes, exponiendo a continuación la nueva estrategia, detallando sus algoritmos de búsqueda por rango y búsqueda kNN, presentando y analizando los resultados obtenidos, para finalizar comentando sus capacidades dinámicas.
- En el capítulo 6 se plantean las conclusiones de la tesis, exponiendo las contribuciones y mencionando algunas líneas de trabajo futuro.
- El apéndice A resume la notación utilizada.
- El apéndice B describe los acrónimos empleados.
- El apéndice C referencia las publicaciones relacionadas con esta tesis, indicando las menciones que se les han otorgado y las citas recibidas.
- Finalmente el apéndice D argumenta las razones de la inclusión de las referencias históricas.

Capítulo 2

Fundamentos de búsqueda en espacios métricos

Un espacio métrico se obtiene al dotar a un conjunto de una función de distancia que cumple tres propiedades: identidad, simetría y desigualdad triangular. El concepto de distancia es una de las primeras nociones matemáticas desarrolladas por la humanidad. Las búsquedas por similitud o por proximidad tratan de determinar los objetos más parecidos o similares a otros dados, ampliando así el concepto de búsqueda por igualdad, que pasa a ser un caso particular de la búsqueda por similitud. La aplicación de los espacios métricos a las búsquedas por similitud ha permitido establecer un formalismo que, basándose en una función de distancia, establece con rigurosidad la similitud o parecido entre dos objetos y aprovecha sus propiedades, especialmente la desigualdad triangular, para reducir el número de evaluaciones de la función de distancia, ya que este constituye un componente especialmente costoso de la búsqueda por similitud.

En este capítulo se revisan inicialmente los principales planteamientos geométricos que se originaron a lo largo de la historia al considerar el concepto de distancia. La sección siguiente se centra en formalizar los espacios métricos, discutiendo las características de diversas distancias que se utilizan en ellos. Posteriormente se describe la búsqueda por similitud, estudiando los diferentes tipos de consultas que pueden presentarse en ella. Seguidamente se realiza una revisión de las estrategias utilizadas en la búsqueda por similitud sobre espacios métricos, haciendo hincapié en los métodos basados en pivotes y en los métodos basados en particiones, y se afrontan los problemas existentes al considerar grandes dimensiones.

2.1. Distancias y geometrías

La idea que subyace en la realización de una búsqueda por similitud es la de poder concretar lo que subjetivamente percibimos como el parecido o la proximidad entre dos o más elementos, y asignarle un valor objetivo determinado. Por contraposición se utiliza el término distancia para indicar el espacio que hay entre dos cosas, el intervalo de tiempo entre dos sucesos, la diferencia entre unas cosas y otras, e incluso el desafecto entre personas.

La utilización de una distancia fue la base de la geometría euclidiana, en la que se estudian las propiedades del plano y del espacio tridimensional, considerando que la distancia entre dos puntos es la longitud numérica del segmento de recta que los une. Euclides de Alejandría, que vivió hacia el año 300 a.C., determinó en su libro *Los elementos* un sistema de axiomas formado por cinco postulados que son el origen de toda la geometría que lleva su nombre y que durante más de 2000 años determinó el conocimiento geométrico que fue aplicado a la totalidad de las ciencias.[1]

Aunque algunos de sus postulados crearon controversia desde un principio, la trascendencia de la obra de Euclides es abrumadora. Hasta la primera mitad del siglo XVII no se produjo una aportación realmente importante a las ideas iniciales de Euclides, que sirvió además para añadirles consistencia formal, que consistió en la introducción por Descartes[2] del concepto de coordenadas con sus ejes cartesianos y que dio origen a la geometría analítica.

La geometría euclidiana está realmente determinada por el quinto postulado de Euclides, que fue siempre el más cuestionado, y que indica que por un punto exterior a una recta solo se puede trazar una única paralela a dicha recta. A lo largo del tiempo, numerosos matemáticos estudiaron las características de este postulado, incluyendo su independencia de los restantes [Rosenfeld, 1988].

En 1733, el año de la muerte del sacerdote jesuita Giovanni Saccheri, se publica su trabajo referente al quinto postulado de Euclides, en el que partiendo de su cuadrado

[1] A la muerte de Alejandro Magno su imperio fue regentado por algunos de sus generales, entre ellos Ptolomeo I, que primero gobernó y luego reinó en Egipto. Fue el fundador de la biblioteca de Alejandría y entre los sabios a quienes llamó para ese proyecto estaba Euclides, matemático griego que fue capaz de condensar el saber geométrico en un texto cuyos planteamientos han pervivido hasta la revolución geométrica ocurrida en el siglo XIX.

[2] René Descartes fue un gran filósofo y científico que en su niñez tuvo una salud precaria, por lo que pasó muchas horas en cama que aprovechó para leer y meditar, costumbre que mantuvo durante toda su vida. Su sistema de coordenadas lo ideó durante su etapa militar, mientras combatía la inactividad castrense con su raciocinio, parece ser que cuando estaba en su cama observando el vuelo de una mosca y se percató de que la posición del animal podía determinarse mediante tres valores que indicaban las distancias a las paredes y al suelo.

birrectángulo, plantea las hipótesis del ángulo obtuso, que descarta inicialmente, y la del ángulo agudo, que rechaza más por planteamiento moral que científico.

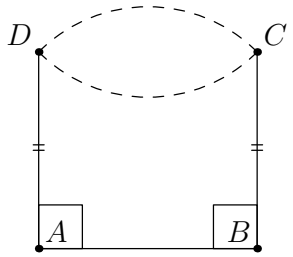


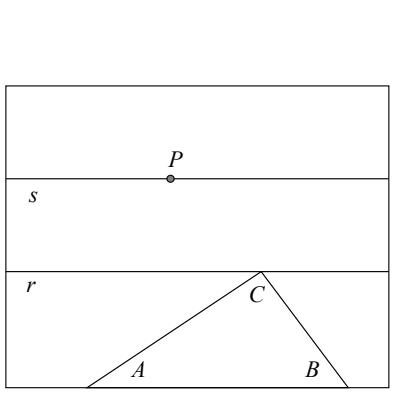
Figura 2.1: El cuadrado de Saccheri.

En el siglo XIX se formalizaron las ideas que contradecían el quinto postulado, dando origen a la geometría hiperbólica, debida principalmente al matemático húngaro János Bolyai y al ruso Lobachevski. De esta forma se pasa de la geometría euclidiana asociada al espacio físico en el que se utiliza la distancia usual y que posee curvatura cero, a la hiperbólica, que puede desarrollarse sobre la pseudo–esfera, presenta curvatura negativa y posee más de una recta paralela a una dada que pasa por un punto exterior.

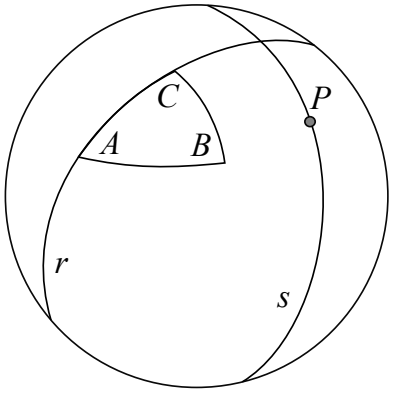
Poco después Riemann[3] desarrolla la geometría elíptica, con curvatura positiva y donde no hay ninguna recta paralela que pasa por un punto exterior a otra dada. Además Riemann generalizó estas geometrías y con ello el concepto de distancia. Desde ese momento la geometría euclidiana perdió relevancia y el estudio de la geometría se ha basado en los espacios geométricos de dimensión n o variedades diferenciales debidos a Riemann. Sus ideas las utilizó Einstein para plantear la geometría del espacio–tiempo en su *Teoría de la Relatividad general*.

En la figura 2.2 se representa el quinto postulado de Euclides sobre las geometrías euclidiana, hiperbólica y elíptica, y el resultado de la suma de los ángulos de un triángulo ABC, que es igual, menor y mayor que 90^o respectivamente [Ibáñez, 2010].

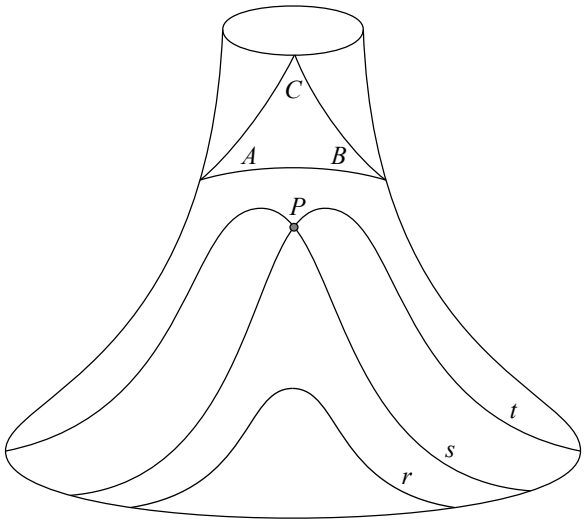
[3]Bernhard Riemann fue un eminente matemático alemán de mediados del siglo XIX, tímido, de salud delicada, a quien le horrorizaba hablar en público y que murió de tuberculosis a los 39 años, pero que revolucionó el estudio de la geometría, gracias fundamentalmente a las ideas expuestas en la conferencia que impartió para su habilitación en la Universidad de Gotinga.



(a) Euclidiana



(b) Elíptica



(c) Hiperbólica

Figura 2.2: El quinto postulado en diferentes geometrías.

2.2. Espacios métricos

2.2.1. Definiciones iniciales

Los espacios métricos son un concepto matemático utilizado para representar alguna característica diferenciadora entre los elementos de un conjunto, mediante un valor cuantitativo, concretamente un número real mayor o igual a cero. Dados dos elementos del conjunto, se denomina *distancia* entre los dos elementos al valor que mide la diferenciación entre ellos. La utilización del término *distancia* proviene de la representación de la distancia entre dos puntos del plano en la geometría euclidiana, aunque en espacios métricos se aplique a ámbitos en los que no se dan las propiedades espaciales propias de dicha geometría.

Un *espacio métrico* es un par (X, d) formado por un *universo* de objetos X y una *función de distancia* o *métrica* $d : X \times X \longrightarrow \mathbb{R}$ que cumple las siguientes propiedades:

- Identidad: $\forall\, x, y \in X, \quad d(x, y) = 0 \Leftrightarrow x = y$
- Simetría: $\forall\, x, y \in X, \quad d(x, y) = d(y, x)$
- Desigualdad triangular: $\forall\, x, y, z \in X,\ d(x, y) \leq d(x, z) + d(z, y)$

Dados dos elementos cualesquiera $x, y \in X$ decimos que el valor de $d(x, y)$ es la *distancia* entre x e y con respecto a la métrica d. Cuando se sobreentiende la naturaleza de la métrica utilizada, suele decirse, en un abuso de lenguaje, que X es un espacio métrico. De la misma forma se utiliza el término *distancia* sobreentendiendo que se trata de una *función de distancia*.

De las tres propiedades se deduce que $\forall\, x, y \in X,\ d(x, y) \geq 0$ como se demuestra en el lema 2.1:

Lema 2.1. *Dado un espacio métrico (X, d) su función de distancia d cumple que*

$$\forall\, x, y \in X,\ d(x, y) \geq 0$$

Demostración.

$$2d(x, y) \overset{\text{simetría}}{=} d(x, y) + d(y, x) \overset{\text{dsg.trg.}}{\geq} d(x, x) \overset{\text{identidad}}{=} 0 \quad \Rightarrow \quad d(x, y) \geq 0$$

□

Para realizar un estudio más detallado de las propiedades, suelen expresarse en una forma equivalente a la ya dada:

P1) $\forall\, x, y \in X,\ d(x,y) \geq 0$
P2) $\forall\, x, y \in X,\ d(x,y) = d(y,x)$
P3) $\forall\, x \in X,\ d(x,x) = 0$
P4) $\forall\, x, y \in X,\ x \neq y \ \Rightarrow\ d(x,y) > 0$
P5) $\forall\, x, y, z \in X,\ d(x,y) \leq d(x,z) + d(z,y)$

Si se incumple P4), la función d sería una *pseudo-métrica*, pero si consideramos que todos los objetos que están a una distancia cero son el mismo, pueden adaptarse todos los resultados de una métrica a la pseudo-métrica. Si lo que no se cumple es la propiedad de simetría, se tendría una *cuasi-métrica*. La desigualdad triangular es la propiedad que resulta más importante para su aplicación en la comparación de elementos y proporciona un interesante resultado adicional, conocido como *reajuste de la desigualdad triangular* [Searcóid, 2007], que proporciona una cota inferior de gran aplicabilidad en las búsquedas por similitud:

$$\forall\, x, y, z \in X, \quad |d(x,y) - d(y,z)| \ \leq\ d(x,z) \tag{2.1}$$

cuya demostración se formaliza gracias al siguiente lema.

Lema 2.2. *Dado un espacio métrico* (X,d) *se tiene que* $\forall\, x, y, z \in X$ *:*

$$|d(x,y) - d(y,z)| \ \leq\ d(x,z) \ \leq\ d(x,y) + d(y,z)$$

Demostración. Aplicando la desigualdad triangular se obtiene:

$$d(x,y) \leq d(x,z) + d(z,y) \ \Rightarrow\ d(x,y) - d(z,y) \leq d(x,z)$$

Mediante la simetría se transforma en:

$$d(x,y) - d(z,y) \leq d(x,z) \ \Rightarrow\ d(x,y) - d(y,z) \leq d(x,z) \tag{2.2}$$

Procediendo análogamente para $d(y,z)$ se tiene que:

$$\begin{aligned} d(y,z) \leq d(y,x) + d(x,z) \ &\Rightarrow\ d(y,z) - d(y,x) \leq d(x,z) \\ d(y,z) - d(y,x) \leq d(x,z) \ &\Rightarrow\ d(y,z) - d(x,y) \leq d(x,z) \end{aligned} \tag{2.3}$$

De 2.2 y 2.3 se deduce la cota inferior:

$$|d(x,y) - d(y,z)| \ \leq\ d(x,z)$$

La cota superior se da de forma trivial aplicando la desigualdad triangular:

$$d(x,z) \ \leq\ d(x,y) + d(y,z)$$

□

Ejemplo 2.1. Las secuencias de caracteres de un alfabeto:

Considerando que una cadena de un alfabeto es una secuencia finita de caracteres del mismo, se tiene que el par formado por el conjunto de todas las cadenas del alfabeto y como función de distancia la *distancia de edición* o *distancia de Levenshtein*,[4] definida como el número de caracteres que hay que añadir, modificar o eliminar para convertir una cadena en otra, es un espacio métrico. Una aplicación de búsqueda por similitud sobre él, sería obtener las cadenas que están a una cierta distancia de otra determinada inicialmente. □

Ejemplo 2.2. Espacios vectoriales:

El conjunto $\mathbb{R}^k$ está formado por vectores que constan de k coordenadas de números reales $(x_1, x_2, \ldots, x_k)$, sobre el que se pueden definir diversas funciones de distancia para convertirlo en un caso particular de espacio métrico, denominado *espacio vectorial*.

Los datos multimedia son un campo de aplicación de los espacios vectoriales para la realización de búsquedas por similitud. Mediante una función de transformación que extrae las características más importantes de los elementos multimedia, se hacen corresponder esas características con vectores de dimensión k, de forma que una búsqueda por similitud en un ámbito multimedia, como por ejemplo la de fotografías o de representaciones tridimensionales de objetos, se reduzca a la búsqueda por similitud en un espacio vectorial de dimensión k. □

2.2.2. Funciones de distancia

Las funciones de distancia representan la manera de cuantificar la proximidad de los objetos en un dominio determinado. Existen diferentes funciones de distancia que pueden aplicarse a cada dominio, estando normalmente muy condicionadas por este último. En casos determinados la función de distancia es una transformación de alguna otra función que no cumplía alguna de las propiedades de la definición, por ejemplo en el caso de las pseudo-métricas y las cuasi-métricas.

En esta sección se describen distintas funciones de distancia aplicables a diversos dominios. Según la naturaleza de los valores que toman, unas son discretas y otras son continuas.

[4] Vladimir Levenshtein es un científico ruso nacido en 1935 que ha realizado importantes contribuciones a la Teoría de la Información y a la Teoría de Códigos de Corrección de Errores, entre ellas el algoritmo relativo a la distancia de edición que desarrolló en 1965. Se graduó en la Universidad Estatal de Moscú en 1958 y desde entonces trabajó en el Instituto Keldysh de Matemática Aplicada en Moscú. En 2006 el IEEE le concedió la medalla Richard W. Hamming.

Distancia de Minkowski

La familia de funciones de *distancia de Minkowski*[5] se denota con L_p y se define sobre espacios vectoriales de dimensión k de la forma:

$$L_p(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{k} |x_i - y_i|^p \right)^{1/p}, \; p \geq 1 \tag{2.4}$$

siendo x_i e y_i las componentes de los vectores $\vec{x}$ e $\vec{y}$ de $\mathbb{R}^k$.

Algunas de estas funciones son casos particulares destacados, como las siguientes:

- *Distancia de Manhattan*: $L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{k} |x_i - y_i|$
- *Distancia euclídea*: $L_2(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{k} |x_i - y_i|^2 \right)^{1/2}$
- *Distancia máxima* o *de Chebyshev*: $L_\infty(\vec{x}, \vec{y}) = \max_{i=1,\ldots,k} (|x_i - y_i|)$

La distancia de Manhattan,[6] denominada también *distancia City Block*, *distancia boxcar* y *distancia del valor absoluto*, representa la distancia entre dos puntos mediante la suma de las diferencias absolutas de sus coordenadas.

La distancia máxima determina que la distancia entre dos puntos viene dada por la mayor diferencia entre sus coordenadas. La coincidencia con el número mínimo de movimientos de la figura del rey en el juego del ajedrez para ir de una casilla a otra, hace que también se le conozca como distancia del tablero de ajedrez, aunque su denominación más habitual se deba al matemático ruso Chebyshev.[7]

Estas distancias se utilizan si las componentes de los vectores son independientes, como ocurre en numerosos experimentos científicos y en estudios de carácter económico.

[5] Hermann Minkowski fue un matemático alemán de origen judío, nacido en 1864 en la ciudad rusa de Aleksotas, que hoy se conoce como Kaunas, en Lituania. A los ocho años su familia volvió a Alemania. Estudió las formas cuadráticas y realizó aportaciones a la Teoría de Números, a la Física Matemática y a la Teoría de la Relatividad. Fue profesor de Einstein y compañero y amigo de Hilbert. Murió en 1909 cuando era profesor en la Universidad de Gotinga.

[6] Conocida así por la alusión a la distribución en forma de rejilla de la mayoría de las calles de la isla de Manhattan, ya que la manera de desplazarse por ellas para ir de un lugar a otro siguiendo el camino más corto, coincide con obtener esta distancia entre dos puntos.

[7] Pafnuty Lvovich Chebyshev nació en 1821 en Okatovo –hoy Akatovo–, un pequeño pueblo al oeste de Moscú, y realizó importantes contribuciones a la Estadística y a la Teoría de Números. Murió en San Petersburgo, por entonces la capital rusa, en diciembre de 1894, después de ser profesor en su universidad durante 35 años.

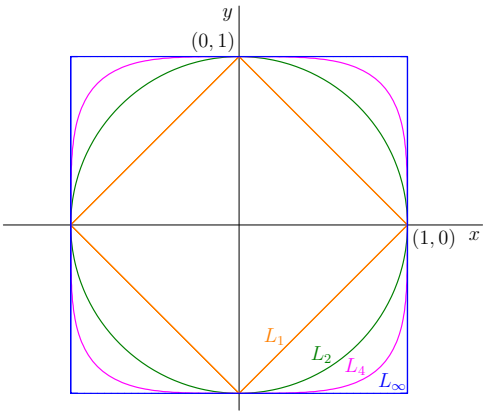


Figura 2.3: Puntos que están a distancia uno del origen de coordenadas aplicando diferentes funciones de distancia de Minkowski L_p.

El valor de la distancia de Chebyshev es consecuencia del siguiente resultado:

Lema 2.3. *Dados $\vec{x}, \vec{y}$ elementos de un espacio vectorial de dimensión k se tiene que:*

$$L_\infty(\vec{x},\vec{y}) \;=\; \lim_{p\to\infty}\left(\sum_{i=1}^{k}|x_i-y_i|^p\right)^{1/p} \;=\; \max_{i=1,\ldots,k}\left(\,|x_i-y_i|\,\right)$$

Demostración. Denotando $M = \max_{i=1,\ldots,k}\left(\,|x_i-y_i|\,\right)$ se tiene que:

$$\sqrt[p]{\sum_{i=1}^{k}|x_i-y_i|^p} \;\le\; \sqrt[p]{kM^p} \;=\; M\sqrt[p]{k} \tag{2.5}$$

Por otra parte se tiene que:

$$\forall\, j \,/\, 1\le j\le k \quad |x_j-y_j| \le \sqrt[p]{\sum_{i=1}^{k}|x_i-y_i|^p} \quad \Rightarrow M \;\le\; \sqrt[p]{\sum_{i=1}^{k}|x_i-y_i|^p} \tag{2.6}$$

De ambos resultados se infiere:

$$M \;\le\; \sqrt[p]{\sum_{i=1}^{k}|x_i-y_i|^p} \;\le\; M\sqrt[p]{k}$$

Tomando límites en los extremos se tiene:

$$\lim_{p\to\infty} M = M \quad ; \quad \lim_{p\to\infty} M\sqrt[p]{k} = \lim_{p\to\infty} Mk^{1/p} = M$$

Aplicando que dadas $f(x) \leq g(x) \leq h(x)$ tales que $\exists \lim_{x\to\infty} f(x) = a$ y $\exists \lim_{x\to\infty} h(x) = a$ entonces $\exists \lim_{x\to\infty} g(x) = a$ se tiene que:

$$\lim_{p\to\infty} \left(\sum_{i=1}^{k} |x_i - y_i|^p \right)^{1/p} = \max_{i=1,\ldots,k} \left(|x_i - y_i| \right)$$

□

Distancia de la forma cuadrática

En determinados entornos las componentes de los vectores presentan alguna relación entre ellas, lo que se traduce en una correlación que la distancia de Minkowski no puede reflejar. Un ejemplo representativo se da al considerar la gama de colores de un conjunto de imágenes, en la que se representa cada color en una dimensión de un espacio vectorial. Para realizar el cálculo de las distancias, deben compararse no solo las componentes del mismo color, por ejemplo la correspondiente al rojo, sino también las que resultan cercanas en la gama, como el color naranja en el caso del rojo. Usando la distancia euclídea, podrían resultar equidistantes del rojo los colores naranja y azul, lo que no se corresponde con la percepción humana de los colores. Por ello se necesita una distancia que refleje la relación existente entre las componentes de los vectores.

Una forma cuadrática en $\mathbb{R}^k$ es una aplicación $q : \mathbb{R}^k \times \mathbb{R}^k \longrightarrow \mathbb{R}$ cuyos valores se obtienen mediante una fórmula del tipo $q(X) = X^T A\, X$ siendo $X = (x_1, \ldots, x_k)^T$ y A una matriz simétrica, esto es, $A = A^T$.

Se establece la denominada *distancia de la forma cuadrática* [Hafner y otros, 1995], aunque ya se había propuesto [Ioka, 1989] y aplicado anteriormente a bases de datos de imágenes [Faloutsos y otros, 1994], como un método de calcular la distancia entre dos vectores k-dimensionales $\vec{x}$ e $\vec{y}$:

$$d_M(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \cdot M \cdot (\vec{x} - \vec{y})}$$

siendo M una matriz $k \times k$ que refleja la relación entre las componentes de los vectores, esto es, cada valor $m_{i\,j}$ está entre 0 y 1 y representa el peso entre las componentes i y j de los vectores $\vec{x}$ e $\vec{y}$ respectivamente.

Distancia de edición

En los ámbitos donde se utilizan cadenas de símbolos para representar los datos, como por ejemplo el reconocimiento de patrones y la biología computacional, se usa la *distancia de edición* o *distancia de Levenshtein* [Levenshtein, 1965], que considera que un alfabeto es un conjunto no vacío de símbolos y una cadena es una secuencia finita de dichos símbolos. Dados dos elementos x e y del conjunto de todas las cadenas definidas sobre el alfabeto, se denota con $x \xrightarrow{k} y$ la conversión de x en y usando k operaciones elementales de inserción, modificación o borrado de símbolos. La distancia de edición entre x e y se denota con $d_E(x, y)$ y es el mínimo valor de k tal que $x \xrightarrow{k} y$, en otras palabras, es el número mínimo de símbolos que hay que añadir, modificar o eliminar para convertir x en y.

Una generalización de la distancia de edición asigna pesos a las operaciones elementales, con lo que se considera la suma de los pesos de las operaciones como el valor de la distancia de edición. Un paso más sería asignar pesos diferentes a distintas transformaciones dentro de una misma operación elemental, por ejemplo que convertir a en b tenga un peso distinto al de transformar a en f. En todos estos casos hay que tener en cuenta la necesidad de cumplir las propiedades de un espacio métrico, especialmente la simetría. Recientemente se ha definido una *distancia de edición contextual* [de la Higuera y Micó, 2008] que normaliza los valores obtenidos a un intervalo entre 0 y 1, asignándolos en función de la longitud de las cadenas.

Distancia de edición arbórea

La *distancia de edición arbórea* [Sankoff y Kruskal, 1983] define una función de distancia entre dos estructuras de árbol, determinada como el mínimo coste necesario para convertir un árbol en el otro, utilizando un conjunto de operaciones de edición sobre árboles como la inserción y el borrado de un nodo. Las operaciones de edición pueden tener el mismo coste en todo el árbol o depender del nivel en el que se producen, por ejemplo diferenciando la introducción de un nuevo nodo cercano a la raíz, de la creación de un nuevo nodo hoja, dependiendo del dominio de aplicación. Diversos trabajos han profundizado en el estudio de esta distancia [Zhang y Shasha, 1989; Bille, 2005]. Al considerar los documentos XML como estructuras arbóreas, se les puede aplicar esta distancia para comparar la estructura de los mismos [Guha y otros, 2002].

Distancias sobre conjuntos

Las distancias anteriores se aplican a elementos de un conjunto, pero hay situaciones en las que lo que interesa es comprobar el parecido que puede existir entre diferentes conjuntos. Un ejemplo sería el caso de distintos usuarios que acceden a unos contenidos, considerando que las referencias de los elementos accedidos por cada usuario forman un conjunto, lo que plantea la posibilidad de estudiar el grado de coincidencia de los intereses de los usuarios.

Siendo A y B dos conjuntos, se define el *coeficiente de Jaccard*[8] como una relación entre el cociente de las cardinalidades de la intersección y de la unión, de la forma:

$$d_J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Dos conjuntos muy parecidos tendrán un valor del coeficiente muy bajo, o lo que es lo mismo, una distancia muy pequeña entre ellos. Utilizando esta distancia se comprueba si dos elementos de dos conjuntos cualesquiera son iguales o no, pero conlleva la pérdida de los matices en los que ambos puntos presentan algún parecido, ya que el coeficiente no puede reflejarlo. Esta circunstancia ocurre normalmente en la comparación de imágenes, en la que una figura es un conjunto de puntos. Para superar la rigidez de este coeficiente, hay que recurrir a otra distancia que permita determinar y cuantificar la cercanía entre dos elementos.

Dados A y B subconjuntos de un espacio métrico (X,d) se define la *distancia de Hausdorff*[9] como

$$d_H(A,B) = \text{máx}\left\{\sup_{x\in A}\inf_{y\in B} d(x,y)\,,\ \sup_{y\in B}\inf_{x\in A} d(x,y)\right\}$$

Dos conjuntos están a una distancia de Hausdorff r si y solo si para cualquier elemento de un conjunto existe algún elemento del otro conjunto que está a una distancia menor o igual a r.

[8] Debido al botánico suizo Paul Jaccard que vivió entre 1868 y 1944, y que estudió la fisiología de las plantas y la biogeografía. Desarrolló el coeficiente para medir la similitud entre conjuntos de muestras.

[9] Felix Hausdorff nació en 1868 en Breslau, ciudad alemana que hoy pertenece a Polonia y que se conoce como Wroclaw. Era de origen judío y fue un gran estudioso de la obra de Nietzsche. Sus aportaciones matemáticas fundamentales fueron en la Teoría de Conjuntos, en la Topología y en la generalización del concepto de dimensión. Se jubiló como profesor en la Universidad de Bonn en 1935, a la edad obligatoria de 67 años, y vivió amargamente sus años de retiro bajo la influencia del antisemitismo, que acabó acosándole, hasta que al llegarle en 1942 una citación para presentarse en un campo de internamiento, decide suicidarse junto a su esposa y a su cuñada, en un intento de preservar la dignidad en medio de la barbarie. La calle de Bonn en la que vivió se denomina oficialmente Hausdorffstrasse desde 1949.

La figura 2.4 representa la distancia de Hausdorff entre dos conjuntos A y B. En negro aparecen en forma de vector las distancias correspondientes a los ínfimos; así para x_6 el $\text{ínf}_{y \in B}\, d(x_6, y)$ es su distancia con x_4, y para x_1, x_3 y x_7 es su distancia con x_2. El supremo de ellas es la distancia de x_7 con x_2, que se resalta uniendo los puntos en naranja. Procediendo análogamente para B, se obtendría que las distancias mínimas de cada punto de B a todos los de A son, para x_4 la de x_6 y para x_2 y x_5 la de x_1. La de x_5 con x_1 es la mayor de ellas y también es la mayor entre los supremos, por lo que es la distancia de Hausdorff entre A y B.

Esta distancia se ha utilizado ampliamente en el tratamiento de imágenes [Huttenlocher y otros, 1993].

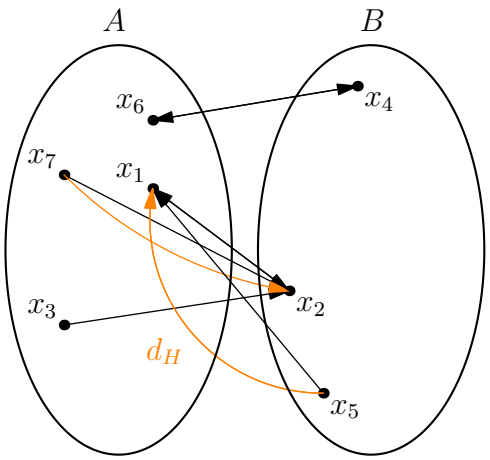


Figura 2.4: Distancia de Hausdorff entre A y B.

Complejidad

El cálculo de una distancia es una operación compleja, ya que supone un coste mucho mayor que la comparación de una palabra clave, como se hace en las estructuras de búsqueda tradicionales.

Las funciones de distancia estudiadas presentan un alto coste computacional, como se muestra en la tabla 2.1, lo que nos lleva a uno de los problemas inherentes a la búsqueda por similitud utilizando espacios métricos, que es el coste del cálculo

de la distancia entre dos objetos, por lo que una de las estrategias fundamentales será intentar reducir el número de evaluaciones de la función de distancia.

Distancia	**Complejidad**
Minkowski	$O(n)$
Forma cuadrática	$O(n^2 + n)$
Edición	$O(n\,m)$
Edición arbórea	$O(n^4)$
Hausdorff	$O(n\,m)$

Tabla 2.1: Complejidad de las métricas estudiadas.

2.3. Búsqueda por similitud

Los métodos tradicionales de búsqueda están orientados al tratamiento de datos que poseen una estructura completamente determinada, generalmente mediante atributos que reflejan las propiedades de los objetos y que conlleva que los objetos se representan mediante registros que se almacenan en bases de datos que siguen un modelo determinado. Las búsquedas habituales en este contexto, en el que los atributos son generalmente de tipo numérico, alfanumérico, de tiempo, etc., están basadas en la búsqueda exacta y sus variantes de búsqueda por rango, que es la que se realiza entre unos valores determinados, pudiendo ser alguno no acotado, y de correspondencia parcial, en la que se restringe la igualdad a unas posiciones determinadas, por ejemplo una cadena alfanumérica que contenga unos caracteres determinados.

Los nuevos tipos de datos utilizados en las aplicaciones actuales, que pueden incluir imágenes, vídeos, audio, documentos de texto, secuencias de ADN, etc., y que generalmente no poseen una estructura completamente determinada, han ampliado las necesidades de búsqueda, ya que la búsqueda exacta es inaplicable en ellos, debido principalmente a las características de los datos involucrados, que, por una parte, presentan una gran dificultad para reproducirse en las mismas condiciones -considérese como ejemplo que dos imágenes del rostro de una misma persona casi nunca son exactamente iguales-, y por otra, los hace muy sensibles a situaciones de ruido, distorsiones, etc. Aparecen así diversos términos como *búsqueda por similitud*, *búsqueda por proximidad* o *recuperación basada en contenido*, para indicar otro tipo de búsquedas que tienen como objetivo determinar los elementos más semejantes, parecidos o cercanos a algún otro.

Determinar la similitud o la semejanza implica una incertidumbre que se ha estudiado tanto en el campo de la psicología como en el de las ciencias de la computación [Santini y Jain, 1999]. Si estas características pueden concretarse de manera rigurosa mediante una función que cumple las propiedades de una función de distancia de un espacio métrico, se tiene un modelo matemático para formalizar el problema de la búsqueda por similitud.

Los espacios métricos se convierten así en un marco teórico que ha dado lugar a numerosos métodos en las últimas décadas. Dado que solo requieren las propiedades de una función de distancia, pueden aplicarse a numerosos problemas de características muy diferentes. Incluso en escenarios en los que es difícil encontrar una función que cumpla los requisitos demandados, pueden efectuarse transformaciones del espacio inicial a otro donde ya sea factible la definición de métricas.

Aunque se ha cuestionado su aplicación debido unas veces a que las funciones de distancia no cumplen todas las condiciones solicitadas, lo que ha originado que diversos trabajos han abordado este problema en los últimos años, y otras por considerar que conllevan una generalización excesiva que las hace aplicables en numerosos casos, pero a la vez que los resultados que obtienen no son los más ajustados en determinados dominios, se reconoce también la solidez de los fundamentos de la utilización de los espacios métricos y los resultados de su aplicación en numerosos dominios [Skopal, 2010].

En este trabajo se considera la búsqueda por similitud en espacios métricos. Se tiene que la *base de datos* o *colección de objetos* sobre la que se realizan las búsquedas es un subconjunto finito $U \subseteq X$ de tamaño $n = |U|$. Una *consulta* se expresa mediante un objeto de búsqueda o de consulta $q \in X$ y un criterio de proximidad a ese objeto. El *conjunto resultado* es el conjunto de objetos de la colección U que cumplen dicho criterio.

2.3.1. Tipos de búsquedas

A continuación se describen varios tipos de consultas que se presentan en la búsqueda por similitud.

Búsqueda por rango

La *búsqueda por rango* es uno de los tipos de búsqueda más intuitivos y puede afirmarse que el más importante porque los restantes pueden realizarse en función de ella [Chávez y otros, 2001b]. El criterio de proximidad está determinado por el *objeto de consulta* o *de búsqueda* q y un *radio* o *rango* r, de forma que la búsqueda

consiste en obtener todos los objetos $x \in U$ que se encuentran a una distancia de q menor o igual a r:

$$R(q,r) \;=\; \{x \in U \,/\, d(q,x) \leq r\} \tag{2.7}$$

Se podría establecer una jerarquía entre los objetos según que su distancia sea más cercana al objeto q. La longitud del radio tiene una incidencia directamente proporcional en la dificultad de realización de la búsqueda. Si la longitud es cero estamos ante el caso particular de la búsqueda exacta sobre q.

La figura 2.5 muestra un ejemplo de búsqueda por rango $R(q,r)$ sobre $\mathbb{R}^2$ utilizando la distancia L_2 en un conjunto $U = \{x_1, x_2, \ldots, x_7\}$. Los elementos de U que cumplen el criterio de búsqueda son los que forman el conjunto $\{x_1, x_2, x_6\}$.

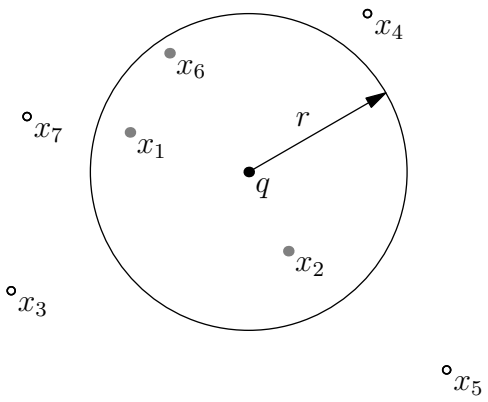


Figura 2.5: Búsqueda por rango.

Búsqueda del vecino más cercano

Aunque la búsqueda por rango es la más elemental, hay situaciones en las que su utilización resulta poco adecuada. Por ejemplo, si se determina un radio muy pequeño, podría no obtenerse ningún resultado; por el contrario, si la longitud del radio es demasiado grande, el resultado podría estar formado por muchos objetos y no resultar representativo. El propio significado del radio, con una longitud determinada, es evidente si se utiliza la distancia de edición, pero resulta confuso al utilizar una distancia euclídea sobre un conjunto de imágenes. En estos casos una alternativa interesante es determinar un número concreto de objetos que sean los más cercanos a otro dado.

Se conoce como *búsqueda del vecino más cercano (nearest neighbor)* la obtención del objeto que está más próximo a uno dado:

$$NN(q) \;=\; \{x \in U \;/\; \forall\, y \;\in U \quad d(q,x) \leq d(q,y)\} \tag{2.8}$$

Búsqueda de los k-vecinos más cercanos

La generalización de la búsqueda del vecino más cercano a k elementos es la *búsqueda de los k-vecinos más cercanos* y consiste en recuperar el subconjunto formado por los k elementos más cercanos a uno dado:

$$kNN(q) \;=\; \{A \subseteq U \;/\; |A| = k,\; \forall\, x \in A \quad \forall\, y \in U - A \quad d(q,x) \leq d(q,y)\} \tag{2.9}$$

La figura 2.6 muestra un ejemplo de búsqueda de los 2-vecinos más cercanos a q, que resultan ser x_1 y x_2.

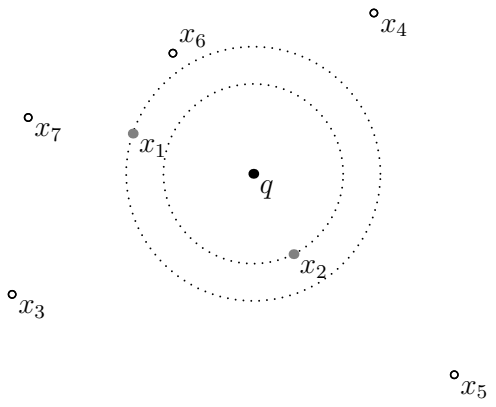


Figura 2.6: Búsqueda de los 2-vecinos más cercanos.

Al indicar un número determinado de vecinos a buscar y dado que puede haber varios elementos a la misma distancia del objeto de búsqueda, puede ocurrir que se obtenga un número mayor de candidatos que los inicialmente estipulados. Esto ocurre si buscamos los 2-vecinos más cercanos a q' en la figura 2.7, ya que además de x_2, que es el más cercano, aparecen x_1 y x_6 a la misma distancia de q', pudiéndose elegir cualquiera de ellos como el segundo vecino más cercano.

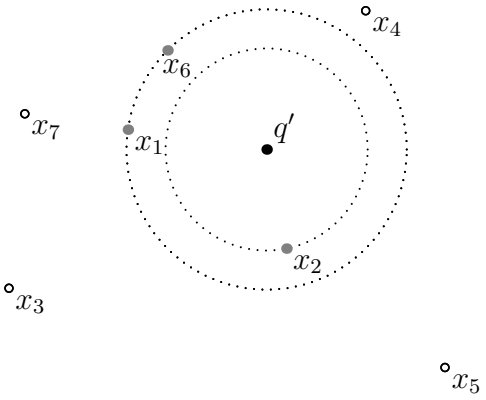


Figura 2.7: Búsqueda de los 2-vecinos más cercanos con coincidencias.

Variantes destacadas

En determinadas situaciones es interesante realizar un tratamiento inverso y encontrar los objetos para los que el objeto de consulta q está entre sus k-vecinos más cercanos. A este tipo de consulta se le conoce como *búsqueda inversa de los k-vecinos más cercanos*:

$$RkNN(q) = \{R \subseteq U \;/\; \forall\, x \in R \quad q \in kNN(x),\ \forall\, y \in U - R \quad q \notin kNN(y)\} \quad (2.10)$$

La figura 2.8 muestra un ejemplo de búsqueda inversa de los 2-vecinos más cercanos.

Mediante este tipo de búsqueda se obtiene una referencia de la situación del objeto de búsqueda en relación a los restantes, ya que ahora se determinan los elementos para los que el objeto de búsqueda es uno de los más cercanos. Este cambio de orientación en la realización de la búsqueda conlleva ciertas particularidades, por ejemplo, que un elemento cercano al objeto de búsqueda pueda no estar en el conjunto de la búsqueda inversa, porque existen otros objetos más cercanos a él de lo que está el propio objeto de búsqueda. Esto ocurre con x_1 en la figura 2.8, ya que es el segundo objeto más cercano a q y a la vez se tiene que q no está entre los 2-vecinos más cercanos de x_1, por lo que $x_1 \notin R2NN(q)$.

De la misma forma un objeto lejano puede formar parte del conjunto de la búsqueda inversa porque, aún así, el objeto de consulta sigue siendo uno de los más cercanos a él. Es lo que le sucede a x_5 en la figura 2.8, ya que tiene a q entre sus 2-vecinos más cercanos a pesar de ser el objeto más lejano a él.

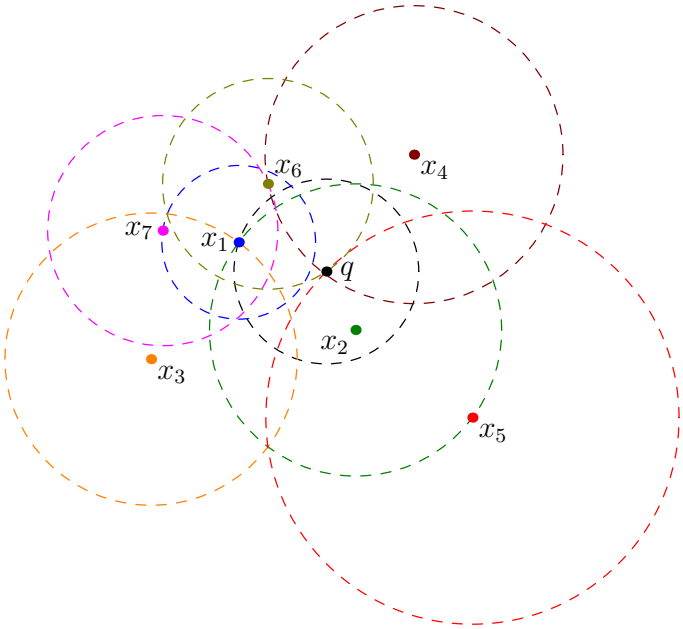


Figura 2.8: Búsqueda inversa de los 2-vecinos más cercanos.

Un caso particular se da si $k = 1$ con lo que la búsqueda inversa obtendría el vecino más cercano.

Desde su reciente introducción [Korn y Muthukrishnan, 2000] se han estudiado diversas aplicaciones de este tipo de búsqueda en diferentes campos.

En algunos problemas, bien porque no se requiere un alto grado de exactitud en el cálculo del vecino más cercano, o bien porque ese cálculo resulta muy costoso en términos de espacio utilizado y de tiempo consumido, se puede considerar la realización de una consulta que obtenga un vecino más cercano pero de manera aproximada, de forma que lo que realmente se obtiene es un punto que está muy cerca del verdadero vecino más próximo.

Este tipo de consulta se conoce como *búsqueda aproximada* y su formalización implica que dado un $\varepsilon > 0$ se dice que p es un *vecino más próximo aproximado* $(1 + \varepsilon)$ de q si:

$$d(p, q) \leq (1 + \varepsilon)\, d(p^*, q)$$

siendo p^* el verdadero vecino mas cercano de q, con lo que p está dentro del error relativo ε del vecino más cercano de q.

Join por similitud

De la misma forma que en bases de datos relacionales el operador de join obtiene las tuplas que coinciden en los valores de atributos que pertenecen al mismo dominio, en espacios métricos se plantea una generalización del mismo, definiendo el *join por similitud* como el conjunto de pares (u, v) que están a una distancia menor que un valor $\varepsilon \geq 0$ con $U, V \subseteq X$ y $u \in U$, $v \in V$:

$$SJ(U, V, \varepsilon) = \{(u, v) \in U \times V \,/\, d(u, v) \leq \varepsilon\}$$

Como caso particular pueden coincidir U y V con lo que se realizan las búsquedas sobre el mismo conjunto, denotándolo como $SJ(U, \varepsilon) = SJ(U, U, \varepsilon)$. Este tipo de búsqueda se usa con frecuencia en minería de datos, en los procesos de detección de duplicados y en la integración de datos.

En la figura 2.9 se muestra un ejemplo de join por similitud sobre $U \subseteq \mathbb{R}^2$ en el que hay tres elementos x_1, x_6 y x_7 que están a una distancia menor entre ellos que el valor elegido de $\varepsilon = 2,4$. Además x_3 y x_7 están también a una distancia menor que ε.

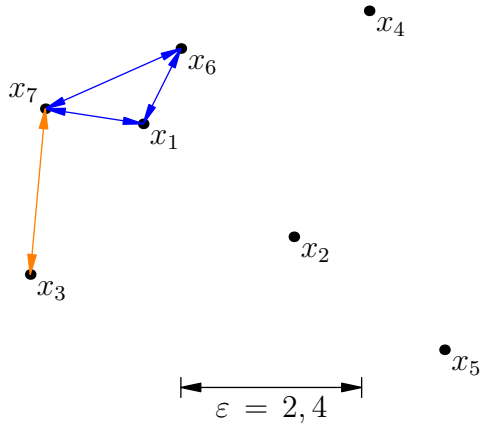


Figura 2.9: Join por similitud sobre el mismo conjunto para $\varepsilon = 2,4$.

2.4. Estrategias utilizadas en espacios métricos

Como ya se ha expuesto al principio de la sección 2.3, en la búsqueda por similitud se plantea la obtención de elementos que no son exactamente iguales al objeto de búsqueda. La manera en la que se determina la similitud se asemeja más a cómo lo hace la mente humana que a otros procedimientos utilizados en computación, en los que, aún bajo condiciones de incertidumbre, se parte de un modelo y se exploran las transformaciones que pueden realizársele hasta convertirlo en el objeto a buscar. Por su parte, la mente humana reconoce la similitud entre objetos diferentes a través de sus características fundamentales, sin realizar una secuencia de transformaciones de uno a otro, sino que se centra primero en destacar una serie de peculiaridades más relevantes, y después en reconocerlas [Patella, 1999]. Los espacios métricos permiten convertir esas características en valores concretos mediante la utilización de una función de distancia.

La manera trivial de implementar la búsqueda por similitud consiste en comparar el objeto de búsqueda con todos los de la colección, en lo que supondría un recorrido secuencial de la base de datos, pero esto no es adecuado porque las funciones de distancia utilizadas originan un coste computacional muy elevado. El problema se incrementa si se trabaja con colecciones de gran tamaño. Por este motivo uno de los objetivos de los métodos de búsqueda en espacios métricos consiste en resolver las consultas reduciendo en lo posible el número de evaluaciones de la función de distancia. Para ello realizan un preprocesado de la colección, construyendo un índice que contiene datos que se utilizan posteriormente en el proceso de búsqueda, para descartar objetos del conjunto resultado sin necesidad de compararlos directamente con el objeto de búsqueda, con lo que se evita el cálculo de la función de distancia.

Los criterios utilizados para descartar objetos y así no tener que compararlos con el objeto de búsqueda, son un elemento clave para reducir el número de evaluaciones de la función de distancia cuando se utilizan métodos de búsqueda por similitud sobre espacios métricos, por lo que más adelante se formalizarán con detalle.

La complejidad de una búsqueda viene dada principalmente por el número de evaluaciones de la función de distancia necesarias para resolver una consulta. Sin embargo, hay otros factores que influyen en el coste total de una búsqueda, como el tiempo de entrada/salida necesario para acceder al índice, que es mayor si la totalidad del índice no puede almacenarse completamente en memoria principal, y también el tiempo de CPU dedicado a procesar los datos del índice y el resto de operaciones. Frecuentemente estos factores adicionales se consideran despreciables, centrándose los métodos en reducir el número de evaluaciones de la función de distancia.

Propiedades a considerar en las búsquedas por similitud

En la realización de las búsquedas por similitud es deseable que los métodos a aplicar posean una serie de propiedades:

Escalabilidad: Consiste en la capacidad para el tratamiento de bases de datos de gran tamaño sin perder calidad en la respuesta.

Independencia de los datos: Las soluciones deben ser aplicables a diferentes dominios, tipos de datos y distribuciones de los mismos.

Dinamismo: Frente a métodos estáticos que solo son aplicables si no hay modificación en los objetos, sería deseable que se permitiese el borrado y la introducción de objetos, tanto para bases de datos existentes como en las que son de nueva creación.

Eficiencia: Indica las consideraciones relacionadas con el coste, en términos de tiempo de CPU y de entrada/salida, que deben ser una parte importante en la evaluación de los métodos, porque las funciones de distancia originan cálculos complejos que inciden en el tiempo de CPU, mientras que el tamaño de las bases de datos y de las estructuras auxiliares necesarias para su tratamiento, causan un incremento importante en las operaciones de entrada/salida.

Planteamiento general de las soluciones adoptadas

Las soluciones adoptadas para resolver las búsquedas por similitud utilizan las propiedades de las funciones de distancia, especialmente la desigualdad triangular, para intentar resolver las consultas sin comparar el objeto de búsqueda con cada elemento de la base de datos, sino solo con una pequeña parte de los mismos. Usan los índices para almacenar los datos relevantes que posteriormente facilitan la realización de las búsquedas, ya que hacen posible descartar elementos durante la comparación porque no cumplen los criterios de la búsqueda.

La utilización de un índice requiere la construcción del mismo, normalmente previa al proceso de consulta, y su almacenamiento, lo que supone un coste inicial que generalmente los métodos no vinculan al proceso de búsqueda. Posteriormente se realiza la consulta, que conlleva recorrer el índice, determinando los elementos que no es necesario comparar con el objeto de búsqueda, para quedarse únicamente con los restantes que sí deben compararse.

Todas las propuestas se basan en la elección de una serie de objetos relevantes de la base de datos, realizada generalmente mediante heurísticas, para representar la totalidad del espacio, con el objetivo de que, una vez almacenados los datos

relevantes en el índice y calculadas las distancias del objeto de búsqueda a dichos elementos, la búsqueda pueda resolverse sin que sea necesario realizar un número elevado de nuevos cálculos de distancias.

Este planteamiento ha originado diferentes algoritmos que pueden clasificarse de diferentes maneras atendiendo a diversas características. En esta tesis se ha utilizado una distribución en dos grandes tipos, los *basados en pivotes* y los *basados en particiones*, también denominados *métodos de clustering* o de *tipo Voronoi*. Su importancia queda patente gracias a los estudios que se han desarrollado acerca de ellos [Chávez y otros, 2001b; Hjaltason y Samet, 2003; Samet, 2006; Zezula y otros, 2006]. A continuación se realiza una descripción general de sus características.

2.4.1. Métodos basados en pivotes

Como ya se estableció anteriormente, la búsqueda por rango se realiza sobre un subconjunto finito o base de datos $U \subseteq X$ de tamaño $n = |U|$, puede expresarse mediante un objeto a consultar $q \in X$ y un criterio de proximidad a ese objeto, se denota mediante $R(q, r)$ y está definida en la fórmula 2.7 de la página 26.

Los métodos *basados en pivotes* seleccionan un conjunto de objetos de U a los que se denomina pivotes $P = \{p_1, \ldots, p_m\}$. Los distintos métodos presentan diversas particularidades, que los hacen adoptar soluciones diferentes, por lo que aquí se expone una visión genérica. Inicialmente se calculan las distancias de cada uno de los pivotes a los restantes elementos x_j de la base de datos, $d(p_i, x_j)$ con $p_i \in P$ y $x_j \in U - P$, y se almacenan en un índice.

Para efectuar una búsqueda por rango $R(q, r)$, se calculan las distancias del objeto de búsqueda a cada uno de los pivotes $d(q, p_i)$ con $p_i \in P$. Para determinar si un elemento de la base de datos $x_j \in U - P$ cumple la condición de búsqueda, en vez de calcular la distancia del elemento x_j al objeto de búsqueda q, se comprueba si el elemento puede descartarse de la lista de candidatos; un criterio a seguir sería utilizar el reajuste de la desigualdad triangular mencionado en la fórmula 2.1 de la página 16, siendo $p_i \in P$ uno de los pivotes:

$$|d(q, p_i) - d(p_i, x_j)| \leq d(q, x_j)$$

con lo que se tiene una cota inferior de la distancia a evaluar $d(q, x_j)$, que viene dada por $|d(q, p_i) - d(p_i, x_j)|$.

Dado que las distancias que intervienen en la cota inferior están ya calculadas, puede determinarse si x_j se descarta o no de la lista de candidatos sin realizar cálculos adicionales, ya que si el rango es menor que el valor de la cota inferior, entonces está garantizado que x_j no cumple la condición de búsqueda:

$$r < |d(q, p_i) - d(p_i, x_j)| \leq d(q, x_j)$$

El proceso se representa gráficamente en la figura 2.10 utilizando el espacio vectorial $\mathbb{R}^2$ con la distancia euclídea. La cota inferior $|d(q,p_i) - d(p_i,x_j)|$ resulta, en este caso, mayor que el rango r, con lo que se garantiza que x_j no cumple la condición de búsqueda y puede descartarse de la lista de candidatos. La brillantez del resultado recae en que no ha sido necesario efectuar nuevos cálculos de la función de distancia, sino que se obtiene mediante la utilización de las distancias almacenadas en el índice, $d(p_i,x_j)$, y las calculadas al comparar el objeto de búsqueda con los pivotes, $d(q,p_i)$ con $p_i \in P$.

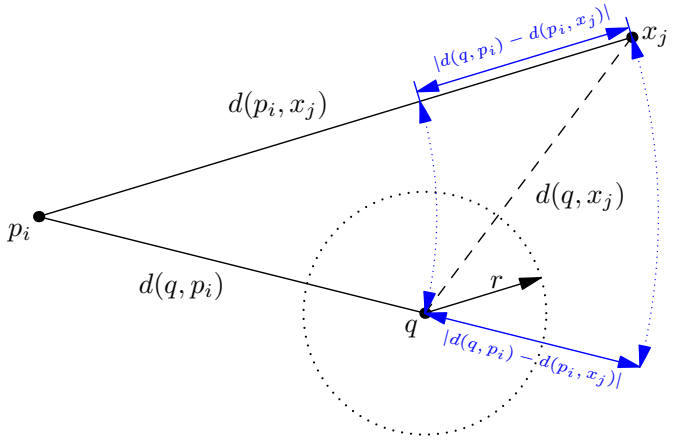


Figura 2.10: Métodos basados en pivotes: Determinando si x_j puede descartarse en la búsqueda por rango $R(q,r)$ utilizando la cota inferior con el pivote p_i.

El proceso seguido tiene la característica de que todos los elementos descartados incumplen la condición de búsqueda, por lo que en este aspecto resulta completo. Podría ocurrir que el valor de la cota inferior fuese menor o igual que el rango, con lo que el elemento no podría descartarse utilizando ese pivote. En este caso el resultado no es determinante y el elemento podría cumplir o no el criterio de búsqueda. Esta situación se representa en la figura 2.11, en la que no puede descartarse x_k debido a que el valor de la cota inferior es menor que el rango, aunque se observa claramente que x_k no cumple la condición de búsqueda.

Esto nos evidencia que dada una búsqueda por rango $R(q,r)$ sobre espacios métricos, para cada pivote p_i existe lo que denominamos una *zona de exclusión* determinada por la distancia del pivote al objeto de consulta, y la suma y la resta, respectivamente, del rango de búsqueda, en la que los elementos que pertenecen a ella no pueden descartarse utilizando ese pivote, independientemente de si cumplen o no el criterio de búsqueda. Si el elemento está fuera de esa zona, puede descartarse

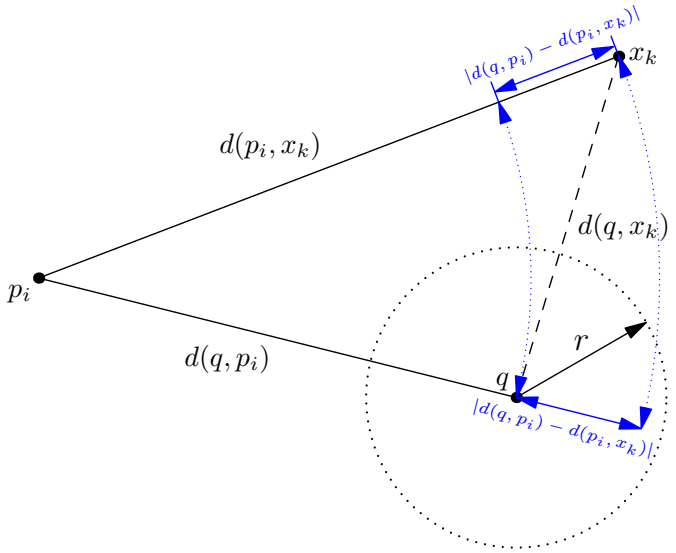


Figura 2.11: Métodos basados en pivotes: Caso en el que un elemento x_k no puede descartarse de la lista de candidatos, aunque realmente no cumple la condición de búsqueda.

con seguridad, pero si está dentro, no se puede descartar utilizando ese pivote. Se tiene así un criterio para descartar un elemento x_k utilizando las distancias ya calculadas, que viene dado por la condición siguiente:

$$\text{Si} \quad d(p_i,x_k) \notin [\, d(p_i,q) - r \,,\, d(p_i,q) + r \,] \quad \Rightarrow \quad x_k \text{ se descarta}$$

En la figura 2.12 se representa este hecho mostrando la zona de exclusión del pivote p_i de manera sombreada, formada por una corona circular con radio inferior $d(p_i,q) - r$ y radio superior $d(p_i,q) + r$. Los elementos que pertenecen a ella no pueden descartarse usando ese pivote. La zona de exclusión contiene a todos los elementos que cumplen el criterio de búsqueda, lo que en la figura se traduce en que la corona circular contiene al círculo que representa la búsqueda por rango.

Si un objeto no se descarta con un pivote, se prueba a descartarlo con otro, ya que se tienen tantas cotas inferiores como pivotes, y por tanto, ese mismo número de oportunidades para descartar cada elemento. Cada pivote utilizado origina su propia zona de exclusión, con lo que a medida que se van aplicando diferentes pivotes, disminuye la zona total de exclusión para cada objeto, ya que es la intersección de todas ellas, lo que origina una reducción del número de elementos en la lista

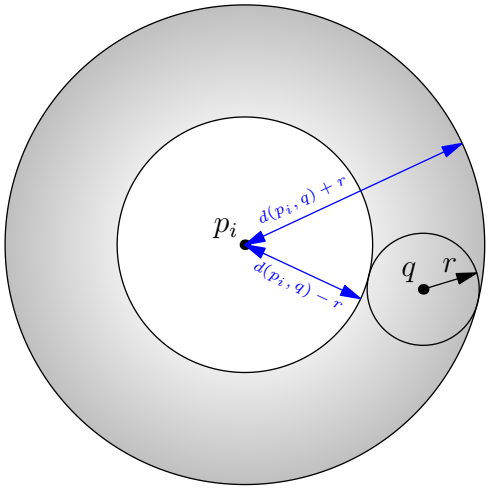


Figura 2.12: Zona de exclusión para el pivote p_i .

de candidatos. Cada zona de exclusión contiene siempre a todos los elementos que cumplen el criterio de búsqueda, pero a la vez la utilización combinada de ellas permite descartar con seguridad más elementos. Todo ello se realiza utilizando las distancias ya calculadas.

La figura 2.13 muestra de forma sombreada la zona de exclusión resultante utilizando tres pivotes, p_1, p_2 y p_3 , sobre la búsqueda por rango $R(q, r)$. Dicha zona está formada por la intersección de las zonas de exclusión de cada pivote, esto es, por la intersección de las tres coronas circulares. En este ejemplo los pivotes elegidos reducen drásticamente los elementos de la lista de candidatos, porque la zona en la que no pueden descartarse los elementos, ocupa una superficie solo un poco mayor que la que le corresponde al círculo que representa la búsqueda por rango. La figura 2.14 amplía el detalle de la zona de exclusión, mostrando claramente la intersección de las tres coronas circulares, en las que están incluidos sus bordes, y cómo contiene al círculo de la búsqueda por rango.

Solo si después de involucrar en las comparaciones a todos los pivotes resulta que un objeto sigue sin poder descartarse, es cuando hay que calcular la distancia entre él y el objeto de búsqueda, teniendo así que realizar un nuevo cálculo de la función de distancia.

Los métodos basados en pivotes se diferencian fundamentalmente en una serie de criterios, como son la manera de elegir los pivotes, el número de pivotes a elegir,

los datos que almacenan y las estructuras de datos que utilizan. En las figuras 2.13 y 2.14 se usan tres pivotes y se resaltan las zonas de exclusión para cada uno y la resultante de su intersección. A partir de los ejemplos gráficos se intuye ya la importancia que algunos de estos criterios tienen en la efectividad de los métodos, como la manera de elegir los pivotes y el número de ellos seleccionado.

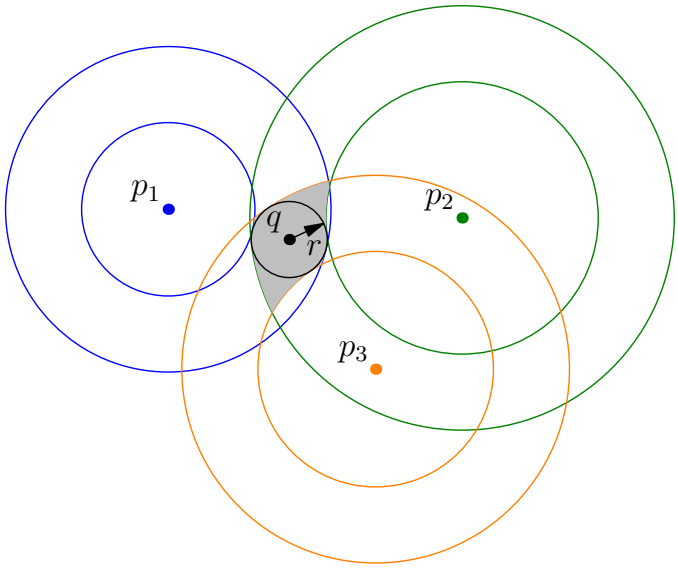


Figura 2.13: Zona de exclusión conjunta para los pivotes p_1, p_2, p_3 .

La manera de elegir los pivotes y el número de los mismos, son dos de los criterios más importantes para que los métodos obtengan buenos resultados. La elección de los pivotes tiene una gran incidencia para determinar la zona conjunta en la que no pueden descartarse los elementos. Por su parte, el número de pivotes utilizado hace que, al aplicarlos a cada elemento para el que hay que evaluar su distancia al objeto de búsqueda, solo resulten efectivos algunos de ellos, haciendo que su efecto combinado convierta en casi irrelevante la inclusión de otros, para un objeto de búsqueda dado. Ocurre también que los pivotes que ante una consulta resultan superfluos, en otra pueden ser determinantes.

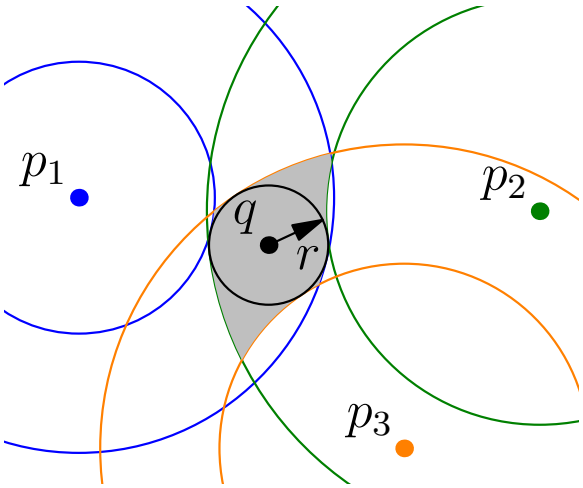


Figura 2.14: Detalle de la zona de exclusión conjunta.

2.4.2. Métodos basados en particiones

Los métodos basados en particiones dividen el espacio en varias zonas formando una partición del mismo, con el objetivo de que en el proceso de búsqueda se puedan descartar completamente algunas de estas zonas, esto es, descartar todos los elementos que pertenecen a ellas, sin necesitar para ello recurrir al cálculo de nuevas funciones de distancia.

En los espacios métricos se utilizan dos criterios de particionado [Chávez y otros, 2001b; Zezula y otros, 2006]: mediante zonas esféricas (*ball partitioning*) o utilizando hiperplanos (*generalized hyperplane partitioning*). En ambos casos se considera un objeto de la base de datos relevante para cada zona, denominado *centro* –para diferenciarlo de los pivotes–, en función del cual se determinará el resto de elementos que pertenecen a la zona, basándose, en general, en considerar que un objeto pertenece a una zona si su distancia al elemento elegido como centro de la misma cumple algún criterio de proximidad.

Utilizando zonas esféricas se elige un objeto como centro y se establece un *radio de cobertura*, que será la mayor distancia del centro a otro objeto de la misma zona, con lo que se obtiene una partición de dos regiones, la esférica determinada por su centro y el radio, y el resto del espacio. El número de regiones puede incrementarse si se usan varios centros, bien desde el inicio o bien si cada región se divide recursivamente.

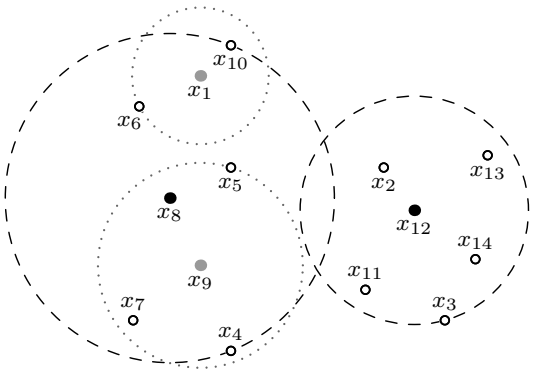


Figura 2.15: Particionado mediante zonas esféricas.

En la figura 2.15 se muestra un particionado en el que inicialmente se usan dos zonas esféricas de centros x_8 y x_{12}, dividiendo a su vez la de la izquierda en otras dos de centros x_1 y x_9.

En el método de los hiperplanos se eligen m objetos c_i, siendo cada uno el centro de una región C_i formada por los objetos que están a una distancia de c_i menor que a los centros de las restantes zonas. Un hiperplano determina la frontera entre dos zonas. El espacio se divide según el conjunto de hiperplanos.

Este último criterio supone, en el caso de espacios vectoriales, crear el diagrama de Voronoi[10] asociado a los centros elegidos, de forma que las regiones generadas, determinadas por su proximidad a cada uno de los centros, son las zonas que forman la partición. En la figura 2.16 se muestra la partición determinada por los hiperplanos al elegir como centros el conjunto de puntos $\{x_1, x_2, x_3, x_4\}$.

[10]Georgy Voronoi fue un matemático ruso que vivió en la segunda parte del siglo XIX y en los primeros años del XX. Un *diagrama de Voronoi* para un número finito de puntos se crea a partir de las mediatrices de los segmentos que unen pares cercanos de esos puntos. Se conoce también como *teselación de Dirichlet*, debido a las teselas o piezas de los mosaicos romanos y en honor al matemático alemán Johann Dirichlet que fue el primero en estudiarlas, y también se le denomina *polígonos de Thiessen*, en recuerdo del meteorólogo estadounidense Alfred Thiessen que vivió desde finales del siglo XIX hasta mediados del siglo XX, y que fue pionero al aplicarlos a la predicción meteorológica como método de división de las zonas terrestres.

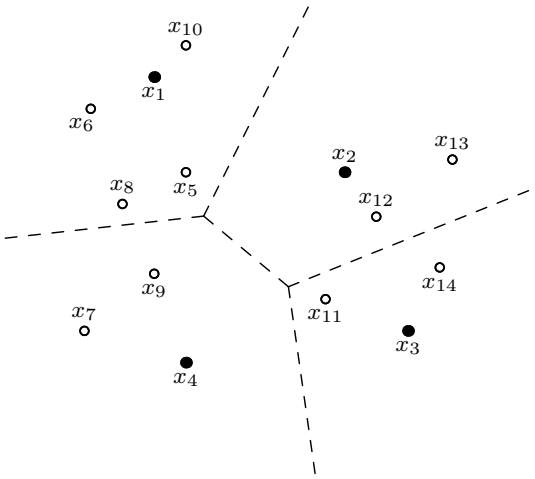


Figura 2.16: Particionado mediante hiperplanos.

Independientemente del método de particionado utilizado, se tienen m centros c_i correspondientes a cada zona o cluster C_i[11] que forman la partición[12] del espacio $U \subseteq X$. En cada zona C_i están los objetos más cercanos al centro de la misma, c_i. Dada una consulta por rango $R(q, r)$, se calculan las distancias del objeto q a cada uno de los centros c_i. Mediante estas distancias, el rango de la consulta, la utilización de un índice con información sobre cada zona y, en su caso, el radio de cobertura, los métodos basados en particiones utilizan en la búsqueda alguna cota que permita descartar una región sin comparar el objeto a buscar con los elementos de la misma.

En líneas generales la idea, que se observa con facilidad en el espacio vectorial $\mathbb{R}^2$, es que como los puntos que cumplen el criterio de búsqueda determinan un círculo, si una región no interseca a ese círculo, entonces ninguno de los puntos que pertenecen a ella formará parte del resultado de la consulta, con lo que la región puede descartarse en su totalidad.

Aunque la formalización del criterio de descarte se realiza posteriormente, se adelanta ya la idea en la que se basa. En el caso del particionado mediante zonas

[11] La literatura anglosajona utiliza el término *clustering* para referirse a los métodos basados en particiones, mientras que a las zonas que definen se les denomina *clusters*. Estos vocablos están plenamente aceptados en la literatura científica en castellano, apareciendo en las actas de numerosas reuniones y congresos, razones por la que se utilizan en esta tesis.

[12] Por abuso de lenguaje a cada zona se le suele denominar también *partición*.

esféricas, el criterio para descartar una zona C_i de centro c_i y radio de cobertura r_{c_i} es:

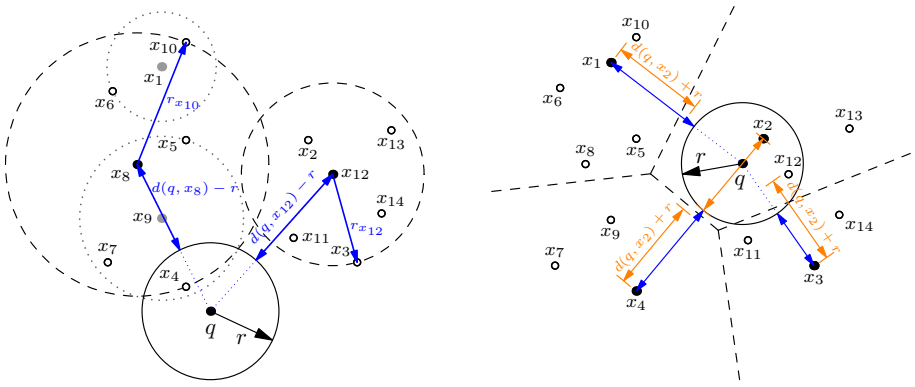
$$d(q, c_i) - r_{c_i} > r \qquad \text{o de otra forma} \qquad d(q, c_i) - r > r_{c_i} \tag{2.11}$$

El lado izquierdo de la figura 2.17 muestra un ejemplo en el que la zona de centro x_8 no resulta descartada y la de centro x_{12} sí.

Si el particionado es mediante hiperplanos, el criterio de descarte de una zona C_i de centro c_i, siendo c el centro más cercano a q, se basa también en que el círculo de búsqueda no interseque a la zona C_i y viene dado por:

$$d(q, c_i) - d(q, c) > 2r \qquad \text{o} \qquad d(q, c_i) - r > d(q, c) + r \tag{2.12}$$

El lado derecho de la figura 2.17 muestra un ejemplo en el que solo se descartan las zonas de centro x_1 y x_4.



(a) Zonas esféricas (b) Hiperplanos

Figura 2.17: Criterio para descartar zonas.

Formalización del criterio de descarte usando zonas esféricas

Si se usan zonas esféricas, la formalización viene dada por el siguiente lema:

Lema 2.4. *Sean c y p objetos en U tales que c es el centro de una zona esférica de radio de cobertura r_c y p pertenece a esa zona, esto es, $d(p, c) \leq r_c$. Dado $q \in X$ se tiene que:*

$$\max\{d(q, c) - r_c, 0\} \leq d(q, p) \leq d(q, c) + r_c$$

Demostración. Por la desigualdad triangular se tiene que:

$$d(q,c) \leq d(q,p) + d(p,c) \Rightarrow d(q,c) - d(q,p) \leq d(p,c)$$

Como $d(p,c) \leq r_c$ ocurre que:

$$d(q,c) - d(q,p) \leq d(p,c) \leq r_c \Rightarrow d(q,c) - r_c \leq d(q,p)$$

lo que unido a las propiedades de la distancia origina la cota inferior.

Aplicando de nuevo la desigualdad triangular y la definición del radio de cobertura, se logra la cota superior de $d(q,p)$:

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

□

El resultado del lema se representa en la figura 2.18 en la que q es un punto que no pertenece a la zona de centro c y donde se evidencia que para cualquier punto p de la zona, se tiene como cotas de $d(q,p)$ las establecidas por el lema.

Si el punto q perteneciese a la zona de centro c, el resultado es aún más evidente, porque la cota inferior sería cero al ser el valor de $d(q,c) - r_c$ menor o igual que cero, y la cota superior se mantiene de manera trivial.

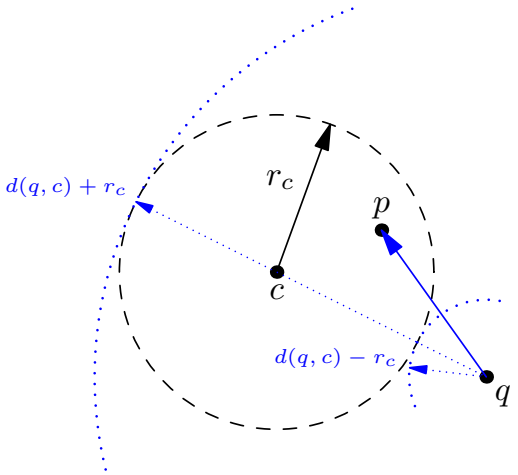


Figura 2.18: Representación del lema 2.4.

En estas condiciones, si se formula una búsqueda por rango $R(q, r)$, la aplicación del lema determina un criterio para excluir las zonas determinadas por el radio de cobertura y los centros, de forma que puede excluirse la zona de centro c y radio de cobertura r_c si ocurre que:

$$d(q, c) - r_c > r$$

Como por el lema se tiene que:

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

se puede garantizar que:

$$r < d(q, c) - r_c \leq d(q, p) \quad \Rightarrow \quad r < d(q, p)$$

lo que implica que el círculo de búsqueda no interseca a la zona de centro c y por tanto dicha zona puede descartarse.

Entonces si en la búsqueda se utilizan los radios de cobertura para eliminar las zonas, por el lema 2.4 el criterio para descartar una zona C_i de centro c_i es:

$$d(q, c_i) - r_{c_i} > r \qquad \text{o de otro modo} \qquad d(q, c_i) - r > r_{c_i} \tag{2.13}$$

siendo r_{c_i} el radio de cobertura de C_i, esto es, la distancia máxima del centro c_i a los puntos de C_i.

En la figura 2.19 se observa que $d(q, x_{12}) - r_{x_{12}} > r$ con lo que la zona esférica de centro x_{12} no interseca al círculo de búsqueda, lo que implica que puede descartarse, ya que está garantizado que ninguno de sus puntos cumplen el criterio de búsqueda. En cambio, en la zona de centro x_8 ocurre lo contrario: la zona interseca al círculo de búsqueda y por lo tanto no puede descartarse. Cuando una zona no puede descartarse, o se subdivide en nuevas zonas, o se calculan las distancias de sus puntos al objeto de búsqueda. Aquí la zona se divide en otras dos, las de centro x_1 y x_9. La primera puede descartarse y la segunda no, ya que $d(q, x_9) - r_{x_9} < r$.

La figura 2.20 permite reflejar el criterio utilizado para descartar las zonas de otro modo, aplicando $d(q, c_i) - r > r_{c_i}$ en vez de $d(q, c_i) - r_{c_i} > r$.

Formalización del criterio de descarte usando hiperplanos

En el caso de utilizar hiperplanos, el siguiente lema permite garantizar una cota inferior:

Lema 2.5. *Sea p un objeto que cumple que $d(c_i, p) \leq d(c, p)$, esto es, el punto p cumple que o es más cercano a c_i que a c o es equidistante entre ambos, entonces dado q se puede establecer una cota inferior de $d(q, p)$ mediante:*

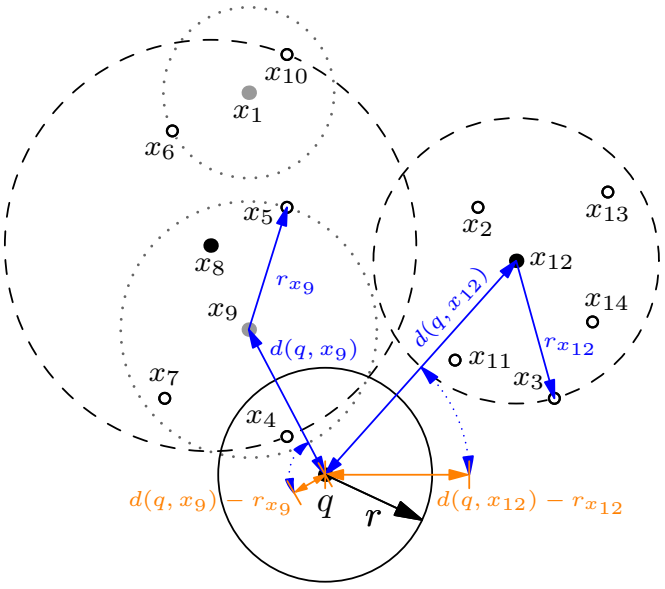


Figura 2.19: Criterio para descartar zonas esféricas.

$$\text{máx}\left\{\frac{d(q,c_i) - d(q,c)}{2}, 0\right\} \leq d(q,p)$$

Demostración. Por la desigualdad triangular y la simetría:

$$d(q,c_i) \leq d(q,p) + d(c_i,p) \Rightarrow d(q,c_i) - d(q,p) \leq d(c_i,p)$$

También por la desigualdad triangular y la simetría:

$$d(c,p) \leq d(c,q) + d(q,p) = d(q,c) + d(q,p)$$

Como además $d(c_i,p) \leq d(c,p)$, se tiene que:

$$d(q,c_i) - d(q,p) \leq d(c_i,p) \leq d(c,p) \leq d(q,c) + d(q,p)$$

y esto determina ya que:

$$d(q,c_i) - d(q,c) \leq 2d(q,p)$$

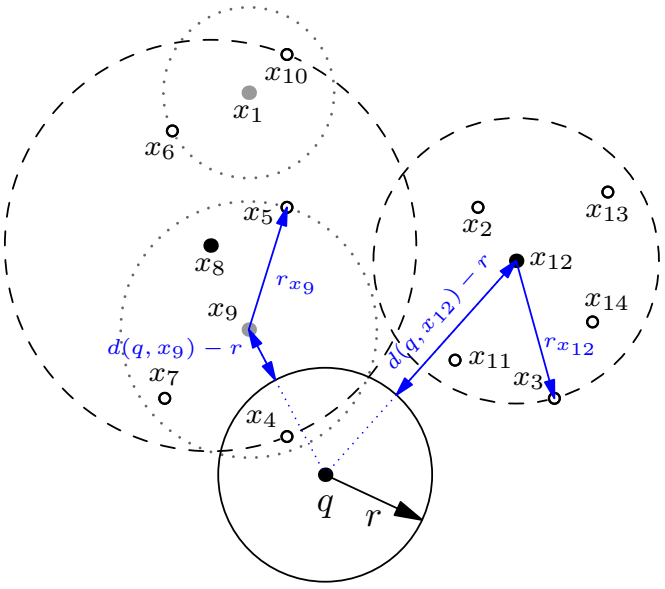


Figura 2.20: Planteamiento alternativo del criterio para descartar zonas esféricas.

Si $d(q,c_i) - d(q,c) > 0 \;\Rightarrow\; d(q,c_i) > d(q,c)$ lo que implica que el punto q es más cercano a c que a c_i. En otro caso $d(q,c_i) - d(q,c)$ sería menor o igual a cero. □

El resultado puede aplicarse a las zonas originadas mediante hiperplanos, siendo los puntos c y c_i centros de dos zonas, y el punto q tal que q es más cercano a c que a c_i.

La figura 2.21 ilustra esta situación. En ella se ha considerado el caso particular de que el punto q esté sobre la recta que une los puntos c y c_i y se observa que para cualquier punto p más cercano a c_i que a c, se cumple que su distancia al punto q es mayor o igual a:

$$\frac{d(q,c_i) - d(q,c)}{2}$$

El lema no impone que el punto q esté sobre la recta que une los puntos c y c_i. En el caso general, mostrado en la figura 2.22, se considera un punto q' que cumple la condición del lema y sea q su proyección sobre la recta que une c y c_i.

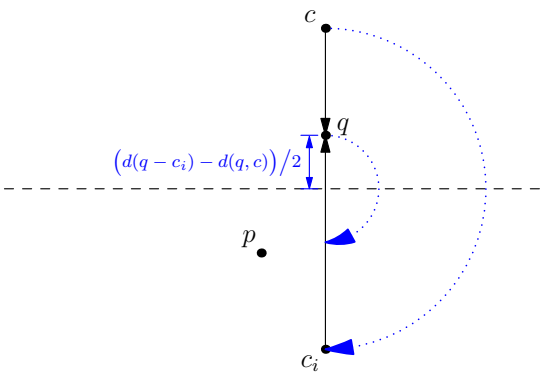


Figura 2.21: Caso particular del lema 2.5.

La distancia de q' a un punto p cualquiera que esté más cercano a c_i que a c, siempre va a ser mayor o igual que la distancia de q' al hiperplano que separa las zonas, esto es, a:

$$\frac{d(q,c_i)-d(q,c)}{2} \tag{2.14}$$

El lema garantiza que la distancia de q' a p va a ser mayor o igual a:

$$\frac{d(q',c_i)-d(q',c)}{2} \tag{2.15}$$

A medida que el punto q' se aleja de la recta que une c y c_i, ocurre que el valor de la expresión 2.15 se reduce[13], como puede observarse en la figura 2.22, por lo que cumplirá el lema de manera más obvia ya que la expresión 2.15 será aún menor que la distancia de q' al hiperplano que separa las zonas, esto es, a la expresión 2.14, resultando finalmente que la expresión 2.14 es una cota inferior para $d(q',p)$.

La negación del resultado del lema permite generar el criterio de descarte para zonas determinadas por particiones basadas en hiperplanos en las búsquedas por rango $R(q,r)$. Se calcula la distancia de q a cada centro de zona c_i, se considera c el centro más cercano a q y la distancia $d(q,p)$ se sustituye por el rango de la consulta r. En estas condiciones, si $d(q,c_i)-d(q,c)>2r$ se tiene que el círculo de búsqueda no interseca a la zona de centro c_i y por tanto dicha zona puede descartarse.

[13]Debido a esto ocurre que se dan situaciones en las que no se cumple el criterio de descarte, pero que tampoco hay intersección no vacía entre una zona y el círculo de búsqueda, como se menciona posteriormente en la página 48.

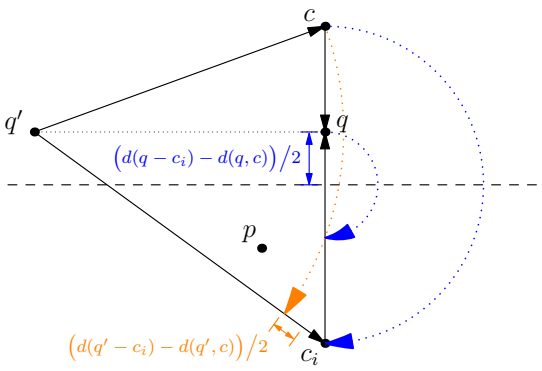


Figura 2.22: Caso general del lema 2.5.

De todo lo anterior se concluye que, si se determinan las zonas en función de los hiperplanos, el criterio para descartar una zona C_i de centro c_i, siendo c el centro más cercano a q, viene dado por el lema 2.5 y garantiza que el círculo de búsqueda no interseca a la zona C_i si:

$$d(q,c_i) - d(q,c) \;>\; 2r \qquad \text{o} \qquad d(q,c_i) - r \;>\; d(q,c) + r \tag{2.16}$$

En la figura 2.23 la partición tiene como centros el conjunto de puntos $\{x_1, x_2, x_3, x_4\}$ y el objeto de búsqueda q se encuentra más cerca de x_2 que de los restantes centros, con lo que $c = x_2$. La distancia entre barras corresponde a $d(q,c) + r \;=\; d(q,x_2) + r$ y es menor que la distancia de q a los centros x_1 y x_4, menos r, esto es:

$$d(q,x_1) - r \;>\; d(q,x_2) + r \qquad \text{y} \qquad d(q,x_4) - r \;>\; d(q,x_2) + r$$

debido a que las zonas de las que ambos son centros, no intersecan al círculo de búsqueda, por tanto pueden descartarse, ya que sus puntos no cumplen el criterio de búsqueda.

La zona correspondiente a x_3 no cumple el criterio para descartarla. Es interesante subrayar que esta zona interseca al círculo de búsqueda pero sin embargo ninguno de sus puntos cumple el criterio de búsqueda, resultando que si hubiera que calcular las distancias del objeto q a todos los puntos de la zona, todas serían mayores que el rango.

El ejemplo permite vislumbrar también la influencia que tiene la elección de los centros y, entre sus consecuencias, el número de elementos que hay en cada

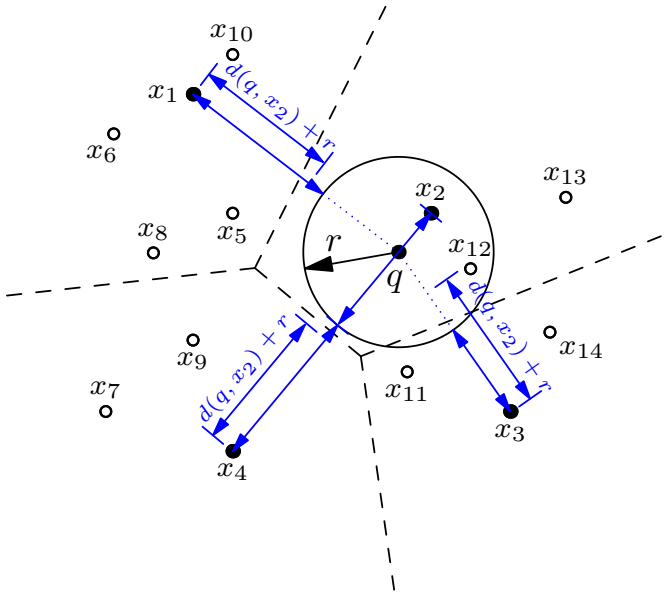


Figura 2.23: Criterio para descartar zonas hiperbólicas ($c = x_2$; $c_i = \{x_1, x_3, x_4\}$).

zona. Si una zona posee muchos puntos y resulta descartada, esto redunda en una mayor eficiencia, pero si esa misma zona no puede descartarse, entonces el número de comparaciones de la función de distancia se incrementa. Cuanto mayor es la zona, mayor probabilidad hay de que no pueda descartarse. Si además casi todos sus puntos incumplen el criterio de búsqueda, se produce un número elevado de comparaciones infructuosas. Si por el contrario se incrementa el número de centros, el tamaño de las zonas se reduce, pero se elevaría el número de comparaciones de la función de distancia.

Si se cumple el criterio para descartar una zona determinada por hiperplanos, se garantiza que no interseca al círculo de búsqueda, pero puede ocurrir que no se cumpla el criterio para descartarla y que tampoco haya intersección entre la zona y el círculo de búsqueda, debido a que $d(q, c_i) - d(q, c)$ se reduce a medida que q se aleja de la recta que une a c y a c_i (ver la página 46), como se percibe en la figura 2.24, donde con varios objetos a buscar y un mismo valor de $r = 0,9$, solo se descarta la zona C_i para la búsqueda de q_0, pero no para las otras dos.

Una situación que puede darse usando tanto zonas esféricas como hiperplanos, es que, ante una consulta determinada, ninguno de los puntos de $U \subseteq X$ cumpla

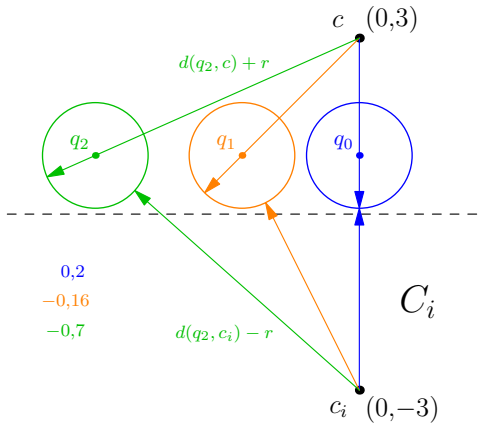


Figura 2.24: Casos extremos del criterio para descartar zonas hiperbólicas.

la condición de búsqueda. Es lo que ocurre en la figura 2.25, en la que además se da la circunstancia de que el círculo de búsqueda interseca a todas las zonas esféricas, a las dos iniciales y a las dos resultantes de dividir la de la izquierda, con lo que ninguna de ellas puede descartarse.

Los métodos basados en particiones utilizan diferentes criterios, tanto en la elección de los centros, como en el particionado del espacio o en la información almacenada en el índice.

2.4.3. Consideraciones acerca del coste

Como ya se ha indicado al principio de la sección 2.4, el principal objetivo de los métodos de búsqueda por similitud en espacios métricos es reducir en lo posible el número de evaluaciones de la función de distancia, ya que se supone que es el componente principal del coste de la búsqueda y que los otros factores ya mencionados, como el tiempo de entrada/salida, pueden considerarse como despreciables. Por este motivo se establece que la complejidad computacional es el número de evaluaciones de la función de distancia necesario para resolver una consulta. La complejidad se divide en dos componentes, complejidad interna y complejidad externa.

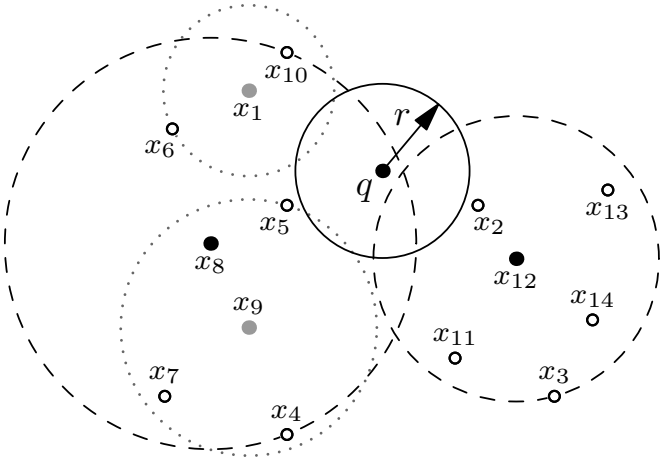


Figura 2.25: Imposibilidad de descartar zonas, aunque sus puntos no cumplen la condición de búsqueda.

Al realizar una búsqueda se explora el índice para determinar los subconjuntos relevantes, para lo que es necesario comparar el objeto de búsqueda con los pivotes o con los centros de los clusters. Al número de comparaciones necesarias para realizar esta función se le denomina *complejidad interna*.

Una vez determinados los subconjuntos relevantes, hay una serie de ellos que no pueden descartarse, por lo que el objeto a buscar debe compararse con los elementos que contienen, calculando la función de distancia entre ellos, para detectar si finalmente cumplen el criterio de búsqueda. Se conoce como *complejidad externa* al número de comparaciones que se llevan a cabo para resolver esta tarea.

Al tratar la eficiencia en la página 32, se ha mencionado que el coste de las búsquedas por proximidad viene determinado fundamentalmente por el número de evaluaciones de la función de distancia, que incrementa el tiempo de CPU, y por el tamaño del espacio que ocupan los datos y los índices, que incide en las operaciones de entrada/salida. El número de evaluaciones de la función de distancia es la suma de la complejidad interna más la complejidad externa.

Los métodos basados en particiones almacenan datos relacionados con cada zona que forma la partición, lo que les permite descartar zonas enteras. Sus necesidades de espacio para almacenar los índices son, en general, lineales, ya que realmente guardan un volumen de datos reducido sobre cada zona y con ello representan ya

todo el espacio; por contra, el número de evaluaciones de la función de distancia que originan es normalmente alto.

Los métodos basados en pivotes suelen ajustar mejor la zona de descarte con respecto al conjunto de puntos que cumplen el criterio de búsqueda, lo que conlleva que el número de evaluaciones de la función de distancia es considerablemente menor que en los métodos de particiones. Esto lo logran mediante la utilización de un índice cuyo tamaño depende del número de pivotes elegido y que almacena un elevado número de valores de distancias de los pivotes al resto de elementos, por lo que el espacio que ocupan es mucho mayor que en los métodos basados en particiones.

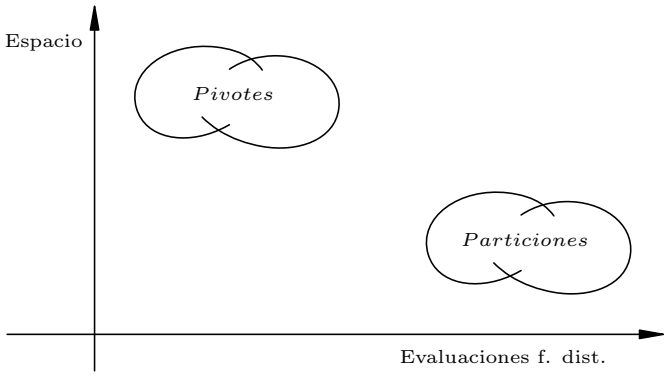


Figura 2.26: Coste de los métodos de pivotes y de particiones.

La figura 2.26 resume gráficamente las consideraciones de coste en ambos métodos.

2.5. Tratamiento de espacios de grandes dimensiones

La efectividad de las técnicas de búsqueda por proximidad en espacios vectoriales está condicionada por la dimensionalidad del espacio. Las búsquedas son más complejas en espacios de dimensionalidad alta, ya que el porcentaje de objetos que los métodos pueden descartar es menor a medida que la dimensión aumenta.

En un espacio vectorial la dimensión del espacio viene determinada por el número de componentes de sus vectores. Pero no en todos los casos se puede trabajar con la dimensión real, como sucede, por ejemplo, en una colección de palabras. En estos

casos se utiliza la denominada *dimensionalidad intrínseca* que es la dimensión en la que se pueden representar los elementos, manteniendo la distancia entre ellos. Por ejemplo, un espacio vectorial de dimensión 45 que se podría representar utilizando únicamente diez dimensiones. Una forma de lograrlo es considerar solo las distancias entre los elementos, sin tener en cuenta las coordenadas.

En [Chávez y otros, 2001b] se ha propuesto una estimación de la dimensionalidad intrínseca para un espacio métrico como:

$$\rho = \frac{\mu^2}{2\sigma^2}$$

con lo que crece con la media (μ) y decrece con la varianza (σ^2). Para ello se utiliza el histograma de las distancias entre objetos en el espacio métrico. Ocurre que si la dimensión intrínseca es alta, los valores en el histograma se concentran alrededor de la media, por lo que su varianza se reduce. Por otra parte para recuperar un porcentaje determinado de la base de datos en una búsqueda por rango, el radio debe ser mayor a medida que crece la dimensión intrínseca, porque los objetos tienden a estar más lejanos unos de otros en estos espacios.

La dimensionalidad intrínseca de un espacio métrico se asocia a la dificultad que conlleva realizar búsquedas por proximidad en él. En el trabajo mencionado se determina que la eficiencia de las búsquedas por proximidad se degrada cuando la dimensionalidad intrínseca es alta. A este fenómeno se le conoce como *maldición de la dimensionalidad*.

Esta situación puede observarse en la figura 2.27 al considerar los histogramas de una búsqueda por rango que utiliza pivotes, en los casos de baja y alta dimensionalidad intrínseca. El criterio para descartar los elementos se reflejó en la figura 2.12 de la página 36:

$$\text{Si} \quad d(p, x_k) \notin [\, d(p,q) - r \,,\; d(p,q) + r \,] \quad \Rightarrow \quad x_k \;\text{ se descarta}$$

Las zonas sombreadas en los histogramas corresponden a los elementos que no se pueden descartar siguiendo el criterio dado, con lo que a medida que la dimensionalidad intrínseca se incrementa, se reduce significativamente el número de objetos que se pueden descartar.

Todo ello conlleva que en espacios con la dimensionalidad muy alta, las estructuras de índices se muestran ineficientes debido a que sufren una dependencia exponencial de la dimensión. Diversos autores han estudiado esta circunstancia [Böhm y otros, 2001], llegando incluso a demostrar que si la dimensión es muy alta, la solución es lineal para un gran número de distribuciones [Beyer y otros, 1999; Shaft y Ramakrishnan, 2006].

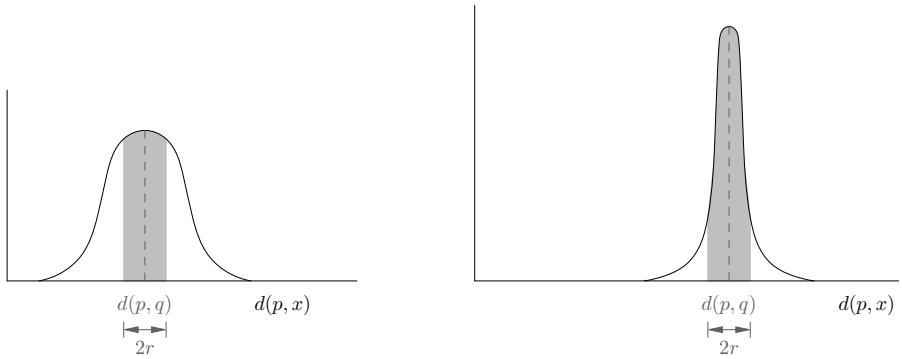


Figura 2.27: Histogramas de distancias con dimensión intrínseca baja (izquierda) y alta (derecha).

2.6. Resumen

En este capítulo han estudiado los fundamentos en los que se basa la búsqueda por similitud en espacios métricos. Inicialmente se ha revisado el enfoque geométrico ligado al concepto de distancia, ya que es el punto de partida de los espacios métricos. Posteriormente se ha formalizado el concepto de espacio métrico y se han expuesto sus características, aportando ejemplos de diferentes distancias. A continuación se han plasmado las ideas fundamentales de la búsqueda por similitud, exponiendo los tipos existentes y sus variantes. Seguidamente se han estudiado las estrategias de realización de búsquedas por similitud en espacios métricos, haciendo hincapié en los fundamentos que les dan consistencia formal, que se han presentado en una serie de lemas, y se ha finalizado con una mención a los problemas existentes en el tratamiento de grandes dimensiones.

Capítulo 3

Soluciones existentes en espacios métricos

La aplicación de los espacios métricos a la búsqueda por similitud ha dado como resultado un amplio conjunto de métodos. Algunos de ellos constituyen los conocimientos básicos de esta temática y son el punto de partida de cualquier acercamiento que se realice sobre ella.

El objetivo de este capítulo es realizar un estudio en profundidad de los métodos más destacados utilizados para efectuar búsquedas por similitud sobre espacios métricos. Se han incluido los métodos más importantes y consolidados, prescindiendo a veces de variantes de los mismos si las modificaciones que presentan suponen solo pequeños matices.

Dada la variedad de métodos existentes y las particularidades que presentan, un problema inicial es realizar una clasificación de dichos métodos, lo que se aborda en la siguiente sección, haciéndolo de manera coherente a lo expuesto en el capítulo anterior e incluyendo también otras propuestas. Luego se analizan los diferentes métodos con ejemplos descriptivos y originales que clarifican su comprensión. Finalmente se expone los argumentos que justifican la necesidad del análisis experimental para evaluar el rendimiento de los métodos

3.1. Clasificación de los métodos

Los métodos de búsqueda por similitud sobre espacios métricos presentan características que a menudo resultan de la aplicación de una combinación de criterios, como por ejemplo que se particione el espacio según una técnica de

hiperplanos pero que se realice la búsqueda usando un radio de cobertura. Algunos requieren que los valores de las distancias sean discretos mientras otros permiten que sean continuos. En unos casos se usan estructuras auxiliares arbóreas y en otros se utilizan arrays; estas estructuras presentan unas veces un tamaño fijo y otras variable. Estas circunstancias originan que se puedan establecer diferentes clasificaciones de los métodos.

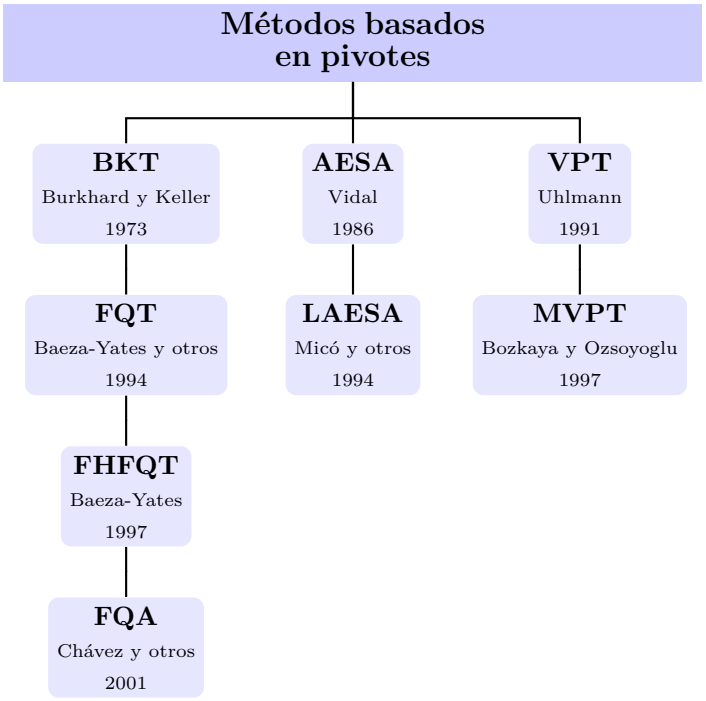


Figura 3.1: Métodos basados en pivotes.

Para establecer una clasificación de los métodos, se ha considerado que el rasgo más determinante es la utilización de pivotes o la realización de una partición del espacio, como ya se ha planteado en el capítulo anterior. Este es el criterio que proponen Chávez, Navarro y otros autores [Chávez y otros, 2001b], y que han seguido otros investigadores [Figueroa, 2007; Pedreira, 2009].

Las figuras 3.1 y 3.2 representan un esquema de los métodos que se van a analizar, considerando los que usan pivotes y los que realizan una partición del espacio, respectivamente. Los acrónimos se describen en el apéndice B.

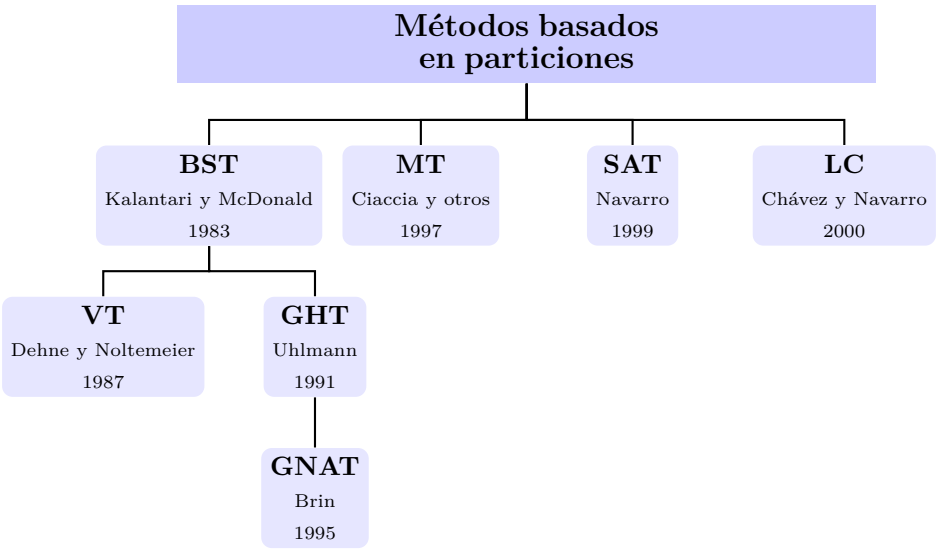


Figura 3.2: Métodos basados en particiones.

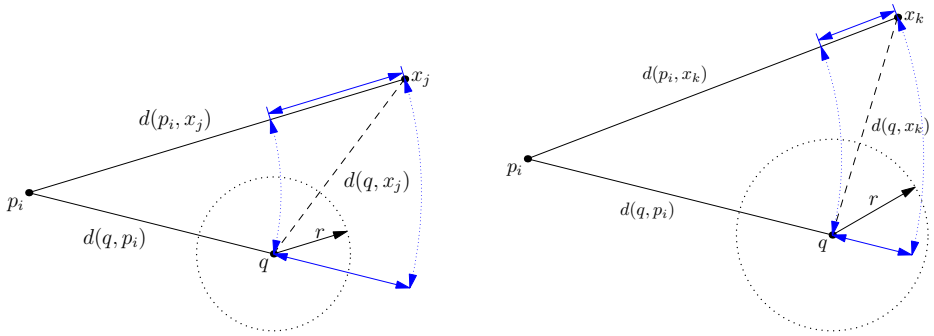
Zezula, Amato y otros autores proponen una clasificación alternativa [Zezula y otros, 2006], e incluso Chávez, Navarro y otros, proponen una clasificación más en la publicación ya mencionada. Estas propuestas se muestran en la tabla 3.1.

3.2. Métodos basados en pivotes

Como se ha descrito en la sección 2.4.1, página 33, el planteamiento general de los métodos basados en pivotes consiste en elegir una serie de elementos de la base de datos denominados pivotes y calcular las distancias de cada pivote a los restantes elementos de la base de datos para almacenarlas en un índice. Para efectuar una búsqueda por similitud, se calculan primero las distancias de cada pivote al objeto de búsqueda y luego se comprueba si los elementos de la base de datos cumplen el criterio de búsqueda, realizando para ello un proceso de descarte de los elementos mediante la utilización de las distancias ya calculadas y la aplicación de las propiedades de la función de distancia. Para cada objeto de la base de datos se utilizan los pivotes necesarios hasta determinar si dicho objeto incumple el criterio de búsqueda y puede entonces descartarse de la lista de candidatos (lado izquierdo de la figura 3.3 en la página 58), de forma que solo los objetos que no han sido descartados en el proceso requieren que se calcule la distancia entre ellos y el objeto de búsqueda (lado derecho de la figura 3.3).

Zezula, Amato y otros		Chávez, Navarro y otros	
Ball partitioning:		**Discrete distance:**	
BKT	FQA	BKT	FQA
FQT	VPT	FQT	
FHFQT		FHFQT	
Hyperplane partitioning:		**Continuous distance:**	
BST		VPT	GNAT
VT		MVPT	VT
GHT		BST	MT
		GHT	SAT
Precomputed distances:		**Others:**	
AESA		AESA	LC
LAESA		LAESA	
Hybrid approaches:			
MVPT	SAT		
GNAT	MT		

Tabla 3.1: Clasificaciones alternativas de los métodos de búsqueda por similitud sobre espacios métricos.



(a) Puede descartarse x_j

(b) No puede descartarse x_k

Figura 3.3: Criterios de descarte de objetos en los métodos basados en pivotes.

Los métodos más destacados basados en pivotes se revisan en los siguientes apartados.

3.2.1. Burkhard-Keller Tree (BKT)

El método *Burkhard-Keller Tree (BKT)* [Burkhard y Keller, 1973] es una de las primeras soluciones para la búsqueda por proximidad sobre espacios métricos.

Considera que la función de distancia es discreta, esto es, que toma solo un número finito de valores diferentes. El índice asociado tiene forma de árbol y se construye eligiendo inicialmente un objeto p como raíz, que será el primer pivote; luego a partir de los posibles valores $i \geq 0$ de la función de distancia, se asignan los restantes objetos de U al correspondiente conjunto $U_i = \{ x \in U \ / \ d(x,p) = i \}$ formado por los elementos que están a distancia i del pivote. Para cada conjunto U_i no vacío se añade un nodo hijo, indicando en la rama el valor de la función de distancia para ese nodo. Finalmente se repite el proceso de manera recursiva para cada nuevo nodo, eligiendo un objeto del mismo que será el pivote del nuevo nodo. La descomposición finaliza cuando se obtienen conjuntos con un número de elementos menor o igual a un valor designado como capacidad de los nodos hoja.

En la figura 3.4 se representa un conjunto de datos sobre $\mathbb{R}^2$ en el que se ha elegido como pivote x_1 y aparecen las distancias existentes entre el pivote y los restantes objetos. Estas distancias, realmente continuas, se han convertido en discretas para adaptarlas a las características del método. El primer nivel del índice se crea partiendo de x_1 y se originan tantos nodos hijos como distancias diferentes hay a los puntos. El resultado se muestra en la figura 3.5.

En el ejemplo se considera que los nodos hoja pueden contener hasta dos elementos, por lo que los nodos que excedan ese número deben descomponerse, siguiendo el mismo criterio que en el nodo raíz. Esto ocurre en los nodos correspondientes a las distancias $4, 5$ y 6. Para ello se elige un elemento como pivote en cada uno de ellos, en este caso x_8, x_{19} y x_{15} respectivamente, y se generan los nodos hijos según las distancia de sus elementos a los nuevos pivotes. En la figura 3.6 pueden verse las distancias para x_8 y para x_{19}, en las que solo hay que considerar los elementos que pertenecen al nodo que se descompone, que aparecen en color gris. De manera análoga se procedería para x_{15} (figura 3.8).

Con todo ello se genera el índice completo del método, que aparece en la figura 3.7.

Para realizar una búsqueda por rango $R(q,r)$ se procede recorriendo el índice, calculando primero la distancia desde el pivote raíz p al objeto de búsqueda q, con lo que ya se determina si p cumple la condición de búsqueda. Además como

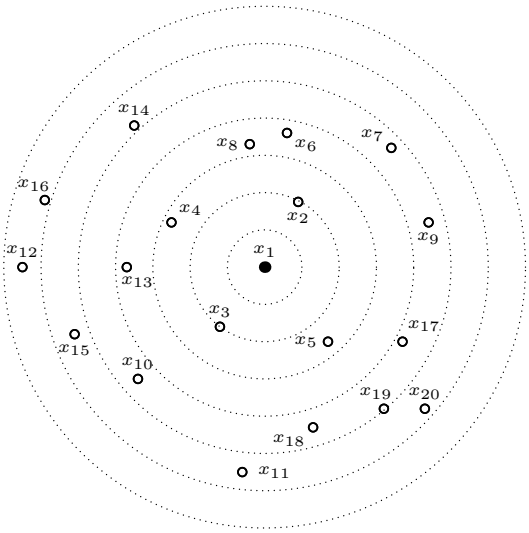


Figura 3.4: Conjunto de elementos a los que se le aplica el método BKT.

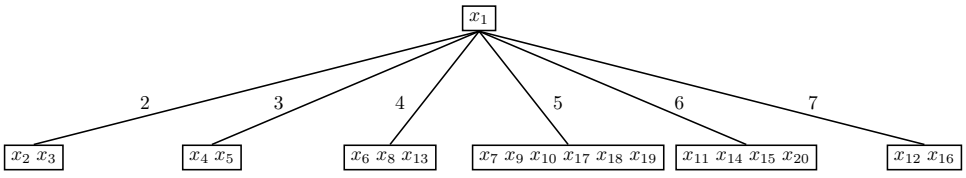


Figura 3.5: Primer nivel del índice en el método BKT.

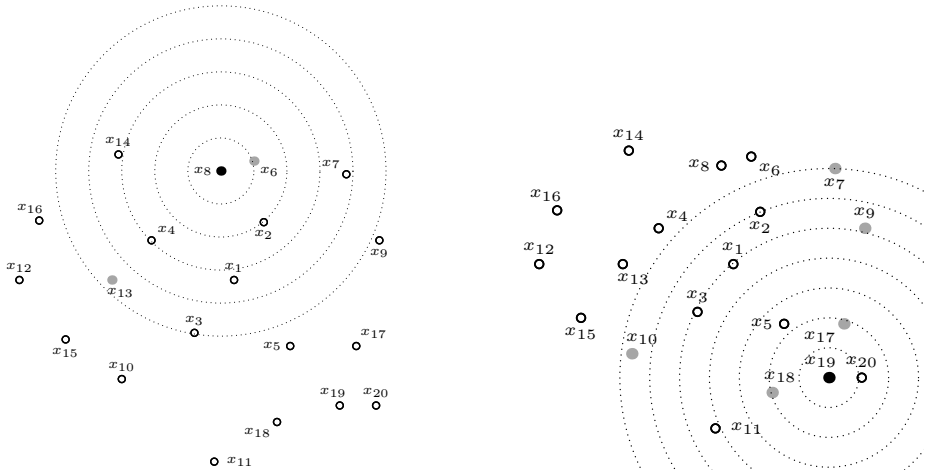


Figura 3.6: Distancias a los pivotes de segundo nivel x_8 y x_{19} en el método BKT.

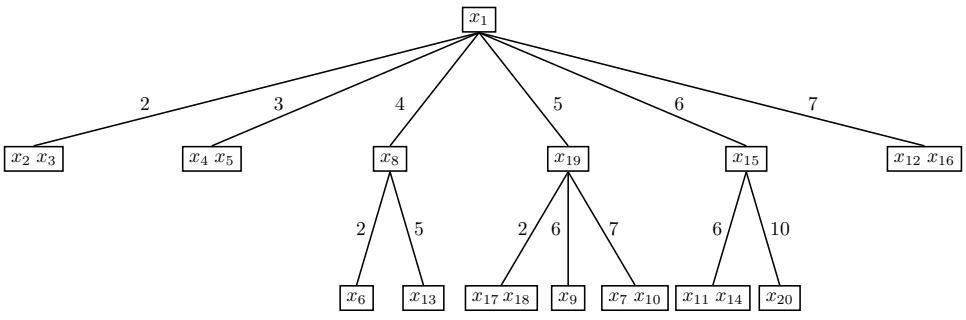


Figura 3.7: Índice completo del método BKT.

consecuencia directa del lema 2.4, habrá que recorrer los nodos cuya distancia asociada i verifique que:

$$\text{máx}\,\{d(q,p) - r\,,\, 0\} \;\leq\; i \;\leq\; d(q,p) + r \tag{3.1}$$

esto es, los nodos para los que la representación de su distancia al pivote raíz tiene intersección no vacía con el círculo de búsqueda, y se procedería recursivamente en ellos, es decir, se calcularía la distancia del pivote del nodo al objeto de búsqueda, y se explorarían los nodos que cumplen la ecuación 3.1. Los nodos que incumplen la condición pueden excluirse, de manera que se desechan todos sus elementos sin calcular su distancia a q; solo es necesario conocer su distancia al pivote raíz –que está almacenada en el índice– y determinar que no cumple las condiciones dadas en la ecuación 3.1.

La figura 3.8 representa el proceso completo de realización de una búsqueda por rango. La subfigura superior izquierda representa el conjunto de elementos, con x_1 como el pivote raíz, y una búsqueda por rango $R(q,r)$. Los nodos hijo que cumplen la condición son los correspondientes a las distancias 6 y 7, por lo que todos los elementos de los restantes nodos pueden desecharse. En el primero de los nodos a recorrer se había elegido como pivote x_{15}, así que se calcula su distancia a q, que resulta mayor que el rango. Luego se comprueba la ecuación 3.1 siendo ahora $p = x_{15}$ y se obtiene que solo se cumple para el nodo que está a distancia 10, como se aprecia en la subfigura superior derecha de la figura 3.8. El nodo resulta ser hoja, por lo que se procede comprobando la distancia de su único elemento x_{20} a q, y resulta que cumple la condición de búsqueda.

El nodo correspondiente a la distancia 7 es un nodo hoja, por lo que también hay que calcular la distancia de q a sus elementos x_{12} y x_{16}, obteniéndose que no cumplen el criterio de búsqueda. Finalmente la figura de la parte inferior representa el recorrido del árbol efectuado para realizar la búsqueda, destacando las ramas que se exploran y los elementos que cumplen el criterio de búsqueda; en este caso, solo x_{20}.

3.2.2. Fixed Queries Tree (FQT)

El método *Fixed Queries Tree (FQT)* [Baeza-Yates y otros, 1994] es una variante de BKT que se diferencia de él por las dos características siguientes:

- Los nodos que están en el mismo nivel utilizan todos el mismo pivote.
- Todos los objetos, incluidos los pivotes, se almacenan en nodos hoja, mientras que los nodos internos se utilizan para modificaciones y para navegar a través del árbol en el proceso de búsqueda.

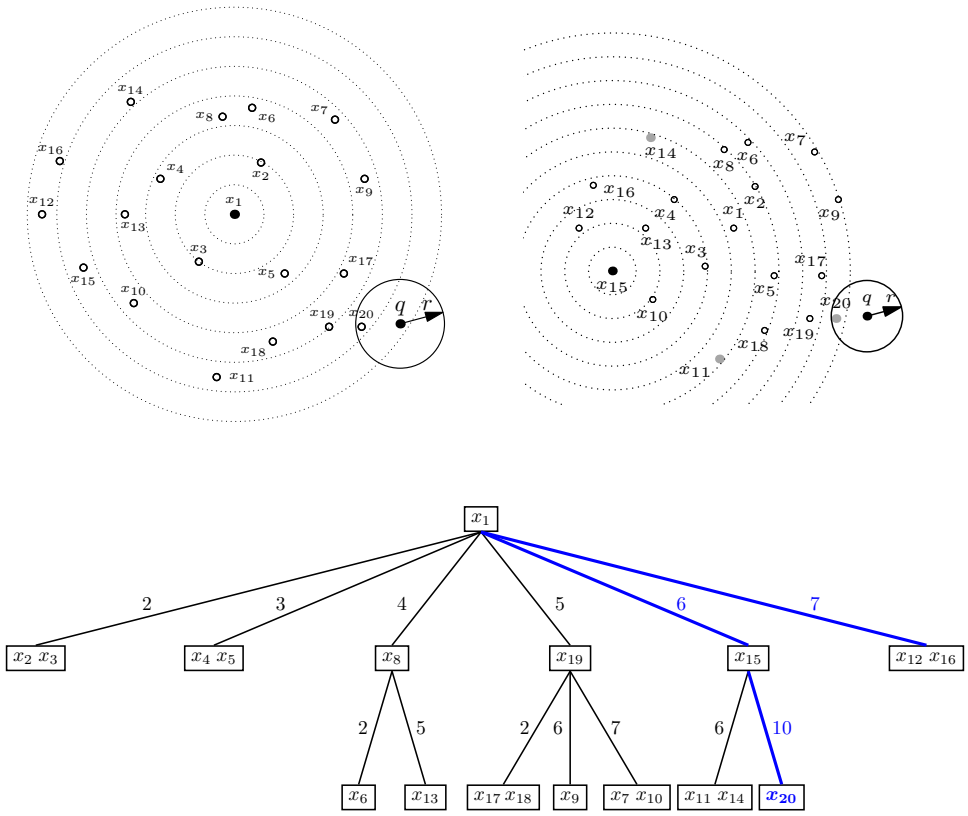


Figura 3.8: Realización de una búsqueda por rango con el método BKT.

Considerando el conjunto de objetos del ejemplo anterior, eligiendo x_1 como el pivote del primer nivel y x_{10} como el del segundo, la figura 3.9 muestra las distancias, el árbol resultante y el proceso de búsqueda por rango.

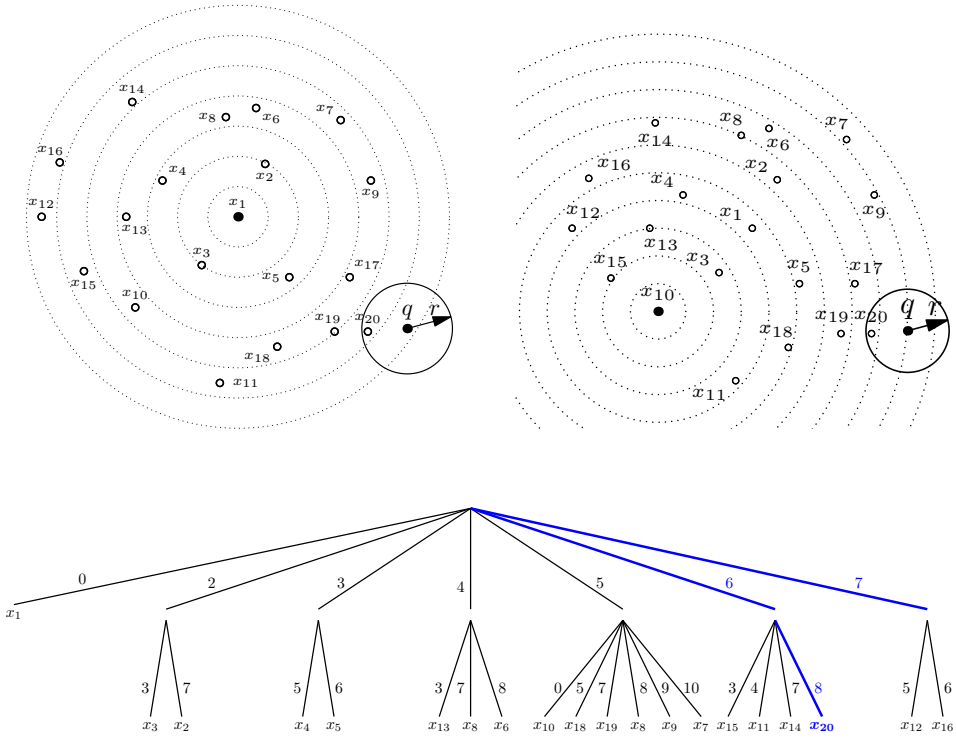


Figura 3.9: Búsqueda en el método FQT con pivotes de primer nivel x_1 y de segundo x_{10}.

El proceso de búsqueda es similar al realizado en BKT, pero FQT aprovecha que todos los nodos que están al mismo nivel usan el mismo pivote, lo que origina una reducción de la complejidad interna. Esto se observa en el ejemplo, ya que para resolver la consulta, en el segundo nivel solo se necesita la distancia de q al pivote de ese nivel, que es x_{10}. Usando x_{10} como pivote, se comprueba mediante la ecuación 3.1 en qué nodos hijo de los dos últimos del primer nivel hay que entrar, resultando que en los que están entre las distancias de 8 y 11. Solo el nodo de x_{20} lo cumple, y como es un nodo hoja, se calcula la distancia de x_{20} a q, resultando además que verifica el criterio de búsqueda.

3.2.3. Fixed Height Fixed Queries Tree (FHFQT)

En el método FQT pueden generarse árboles en los que los nodos hoja están a diferentes niveles. Esa situación se da en el ejemplo considerado, ya que el nodo hoja donde se ubicó x_1 tiene nivel 1 mientras que los restantes tienen nivel 2.

El método *Fixed Height Fixed Queries Tree (FHFQT)* [Baeza-Yates, 1997] es una variante de FQT en la que todos los nodos hoja están al mismo nivel. Esto conlleva que a los nodos hoja que en el método FQT estaban a un nivel inferior a la altura del árbol, se les añaden las ramas necesarias para que sus niveles coincidan con dicha altura.

En la figura 3.10 se representa el árbol del método FHFQT para el ejemplo anterior.

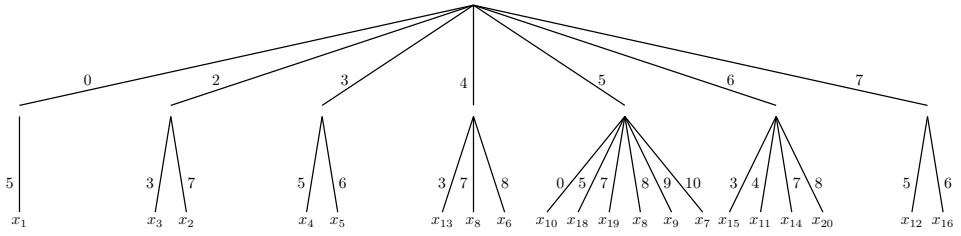


Figura 3.10: Índice del método FHFQT.

La circunstancia de añadir ramas adicionales para que todos los nodos hoja tengan el mismo nivel, puede mejorar el proceso de búsqueda, porque alguno de esos nodos hoja alargados puede descartarse debido a las distancias que presenta frente a otro pivote, con lo que no sería necesario calcular las distancias de los elementos que contiene con respecto al objeto de consulta.

Un ejemplo que ilustra lo comentado se tendría considerando el conjunto de objetos de la figura 3.11, en la que al ejemplo ya tratado, se le añade un x_{21} que está a distancia 8 de x_1, con lo que el árbol tendría una rama más a la derecha.

En el método FQT esa nueva rama llevaría a un nodo hoja de nivel 1, que habría que recorrer y que solo contiene a x_{21}. Al ser un nodo hoja se calcula la distancia de x_{21} a q. Por contra, en el método FHFQT ese nodo hoja aparece en el nivel 2 y su distancia al pivote del segundo nivel, esto es a x_{10}, es igual a 4, con lo que aunque habría que recorrer la rama del primer nivel, no habría que recorrer la del segundo nivel por ser su distancia al pivote de ese nivel menor que 8, con lo que ya no sería necesario calcular la distancia de x_{21} a q.

La igualdad entre el número de pivotes y la altura del árbol permite poder determinar previamente el número de pivotes a utilizar, mientras que en BKT y en FQT el número de pivotes era el resultante de las distancias entre los objetos. El coste del procesado de la estructura se incrementa en este caso, pero se ve compensado por el número de objetos que los pivotes pueden descartar.

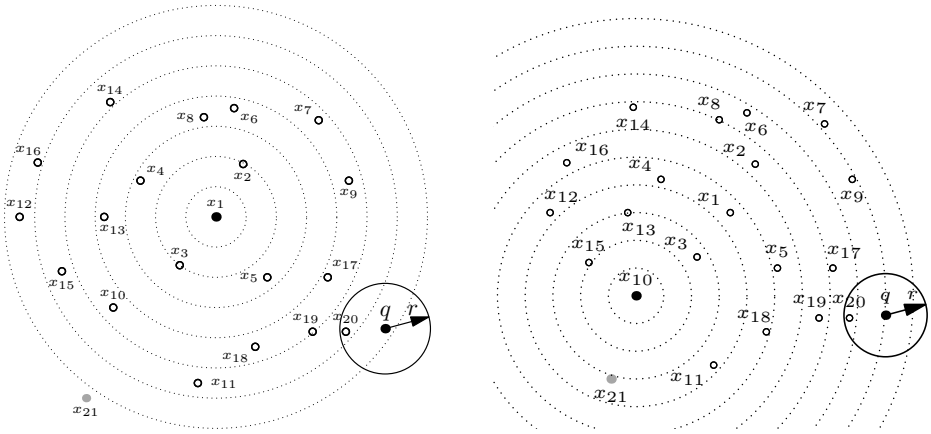


Figura 3.11: Conjunto de objetos con los que FHFQT mejora el proceso de búsqueda.

3.2.4. Fixed Queries Array (FQA)

El método *Fixed Queries Array (FQA)* [Chávez y otros, 2001a] se inspira en el método FHFQT, pero utiliza una estructura de array para la representación de los objetos y de sus distancias a los pivotes.

Dado que en FHFQT todos los nodos hoja están al mismo nivel y hay tantos niveles como pivotes, puede usarse una estructura que almacene los datos que se encuentran al recorrer el árbol de la raíz a las hojas, y de izquierda a derecha, de manera que cada columna corresponda a un nodo hoja, y por tanto a un objeto de la base de datos, y se almacenen en ella las distancias del objeto a cada uno de los pivotes.

La figura 3.12 muestra la estructura resultante en FQA para el conjunto de objetos y el árbol del método FHFQT representado en la figura 3.10. Se observa que los objetos están ordenados primero por su distancia al primer pivote; los que coinciden se ordenan por la distancia al segundo pivote y así sucesivamente.

Objetos	x_1	x_3	x_2	x_4	x_5	x_{13}	x_8	x_6	x_{10}	x_{18}	x_{19}	x_8	x_9	x_7	x_{15}	x_{11}	x_{14}	x_{20}	x_{12}	x_{16}
Nivel 1	0	2	2	3	3	4	4	4	5	5	5	5	5	5	6	6	6	6	7	7
Nivel 2	5	3	7	5	6	3	7	8	0	5	7	8	9	10	3	4	7	8	5	6

Figura 3.12: Array del método FQA.

La comparación de ambas figuras muestra también que cada nodo interno en FHFQT se corresponde con un rango de elementos en FQA. Esto permite que la búsqueda por rango de FHFQT se traslade a FQA realizando una búsqueda binaria.

El método FQA puede usar más pivotes que FHFQT, con lo que se pueden descartar más objetos. Por contra, requiere más tiempo de CPU debido a tener que procesar un índice de mayor tamaño y a realizar la búsqueda binaria. Los autores muestran que FQA supera a FHFQT.

Todos los métodos basados en pivotes presentados hasta el momento se diseñaron para funciones de distancia discretas. Se han propuesto varias maneras de transformar los valores continuos a discretos, mediante la distribución de los posibles valores en una serie de intervalos, que presentan o la misma anchura o el mismo número de objetos [Chávez y otros, 1999, 2001a].

3.2.5. Vantage Point Tree (VPT)

El método *Vantage Point Tree (VPT)* [Uhlmann, 1991; Yianilos, 1993][1] utiliza una estructura arbórea, creada de manera recurrente eligiendo una serie de pivotes que se almacenan en los nodos internos, hasta generar un árbol binario en el que los objetos que no son pivotes se almacenan en los nodos hoja.

El proceso de construcción comienza eligiendo un pivote o *vantage point* $p \in U$, luego se calcula la mediana m de las distancias de p a los restantes objetos de U y se dividen esos objetos en dos subconjuntos de aproximadamente el mismo número de elementos; en el primero están los objetos cuyas distancias al pivote son menores o iguales a la mediana, y en el segundo se encuentran los elementos con distancias mayores o iguales a la mediana.[2]

[1] Aunque algunas publicaciones de la bibliografía referencian a Yianilos como el autor del método, en el trabajo de este se menciona a Uhlmann.

[2] La elección de la mediana de las distancias a p es para que el árbol sea balanceado. La condición de igualdad a la mediana en ambos subconjuntos, se requiere para los casos de coincidencia del valor de la distancia al pivote en varios objetos y que ese valor sea la mediana, con lo que, para que el árbol sea balanceado algunos de esos objetos irán a un subconjunto y al otro los restantes. Esta situación ocurre fácilmente si las distancias son discretas.

El pivote se almacena en la raíz del árbol y los elementos de cada subconjunto formarán parte de cada subárbol. El procedimiento se realiza recursivamente en cada subárbol eligiendo otro pivote, con lo que se obtiene un árbol binario balanceado en el que los nodos internos contienen los pivotes. En los nodos hoja se almacena un número de elementos determinado, en función de su capacidad.

Este método está orientado a la utilización de funciones de distancia continuas, permitiendo también las discretas. El método de asignación de elementos a uno y otro conjunto es un caso particular del criterio de particionado por zonas esféricas o *ball partitioning* ya mencionado en la sección 2.4.2. Una característica diferencial de este método con relación a los analizados hasta ahora, es que no almacena la distancia exacta de los objetos a los pivotes, sino solo un rango en el que se encuentra, que está determinado por la mediana.

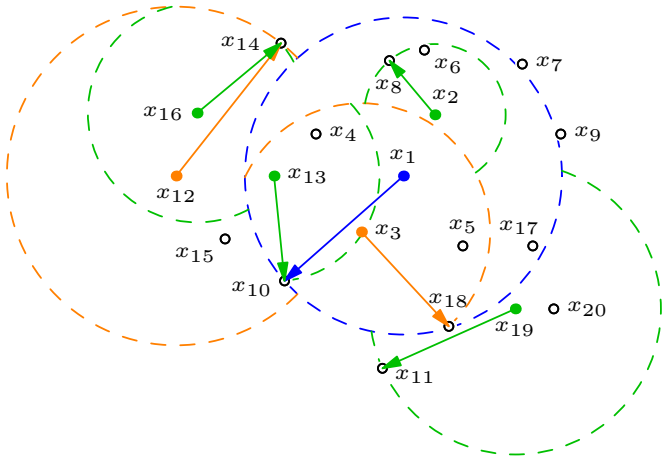
En la figura 3.13 se describe la aplicación del método sobre el conjunto de elementos de los ejemplos anteriores, considerando que la capacidad de los nodos hoja es a la sumo de dos elementos. Se elige como pivote inicial x_1 y resulta que la mediana de las distancias es la correspondiente a x_{10}, generándose la zona de los elementos más cercanos al pivote, que aparece en azul. Entre los elementos de esa zona se elige a x_3 como pivote y se repite el procedimiento. Como los nodos hoja resultantes exceden la capacidad prefijada, se siguen eligiendo pivotes, en este caso x_{13} y x_2, hasta que eso no ocurra. Análogamente se procede con los elementos que quedaron en la zona más alejada al pivote inicial, eligiendo como pivotes primero a x_{12} y luego a x_{16} y a x_{19}, obteniendo finalmente el árbol binario balanceado de la figura 3.13.

El proceso para realizar una búsqueda por rango $R(q, r)$ consiste en recorrer la estructura arbórea desde la raíz hasta las hojas. En cada nodo interno se comprueba si la distancia del pivote p al objeto de consulta q es menor o igual que el rango de la consulta, $d(q, p) \leq r$, en cuyo caso p cumple el criterio de búsqueda.

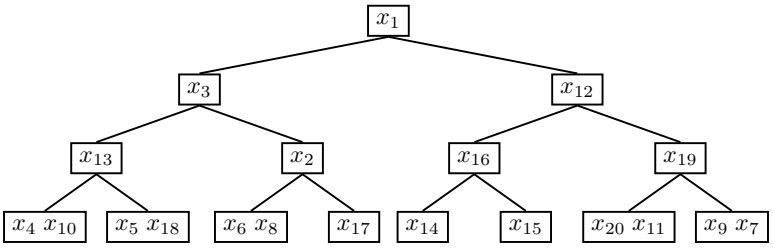
Para establecer el criterio de exploración de los subárboles en cada nodo interno, se necesitan cotas inferiores de las distancias de q a los elementos de cada subárbol.

Para el subárbol de la izquierda, en el que están los elementos cuya distancia al pivote es menor o igual que la mediana, la cota inferior viene dada por la aplicación del lema 2.4 considerando ahora que $c = p$ y que $r_c = m$, lo que garantiza que la distancia de q a cualquier elemento p' del subárbol verifica que $d(q, p) - m \leq d(q, p')$. Si el rango de la consulta es menor que la cota inferior, esto es, si $r < d(q, p) - m \leq d(q, p')$, entonces no es necesario explorar el subárbol. En caso de que no se cumpliese lo anterior, sí habría que recorrerlo, dándose que $r \geq d(q, p) - m$, o lo que es lo mismo, si:

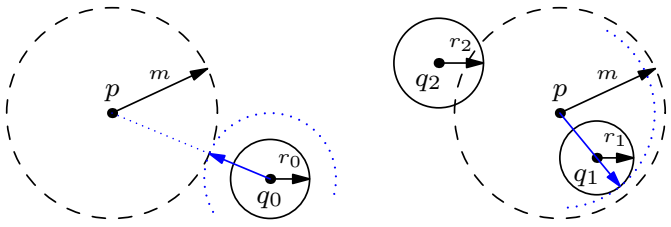
$$d(q, p) - r \leq m$$



(a) Objetos, pivotes y zonas



(b) Índice resultante



(c) Criterio de exploración de los subárboles

Figura 3.13: Aplicación del método VPT.

En el subárbol de la derecha están los objetos situados fuera de la zona de centro p, que sería el pivote del nodo que se explorase, y radio m, que sería la mediana de las distancias. Siendo p' un elemento cualquiera del subárbol, se tiene pues que $m < d(p,p')$. Por la desigualdad triangular $d(p,p') \leq d(p,q) + d(q,p')$, luego $m < d(p,p') \leq d(p,q) + d(q,p')$, y por tanto $m - d(p,q) < d(q,p')$, con lo que $m - d(p,q)$ es una cota inferior estricta de $d(q,p')$. Si el rango de la consulta es menor o igual que la cota inferior, esto es, si $r \leq m - d(q,p)$ no se recorre el subárbol, y habría que explorarlo si no se da lo anterior, esto es, si $r > m - d(q,p)$, con lo que se tiene ya la condición de exploración del subárbol derecho, que se puede expresar de la forma:

$$d(q,p) + r > m$$

En la parte inferior de la figura 3.13 se observan gráficamente ejemplos de los criterios. En la figura de la izquierda se muestra la condición sobre el subárbol izquierdo, resultando que $r_0 < d(q_0,p) - m$ ya que aquí $q = q_0$, con lo que no se explora el subárbol. En la figura de la derecha se refleja la condición del subárbol de la derecha, obteniendo que $d(q_1,p) + r_1 < m$ con lo que no se cumple y por tanto no hay que explorarlo. Se observa también el caso en el que la consulta interseca a ambas zonas, como ocurre con $R(q_2,r_2)$, donde se cumple el criterio para los dos subárboles, por lo que hay que recorrer ambos. Esta situación puede darse en cualquier nodo interno.

En el desarrollo del método se plantea una de las cuestiones cruciales en los métodos basados en pivotes, que es la elección efectiva de los pivotes, considerando que los pivotes más alejados obtienen mejores resultados [Yianilos, 1993].

Se han propuesto varias variantes del método VPT, con planteamientos que van desde utilizar la media en vez de la mediana, lo que puede suponer una mejora del rendimiento en grandes dimensiones pero conlleva que el árbol deja de ser balanceado, influyendo en la eficiencia del método de búsqueda, hasta consideraciones sobre tratar los nodos hoja como cubos con capacidad para varios objetos, que es lo que se ha utilizado en el presente ejemplo.

3.2.6. Multi Vantage Point Tree (MVPT)

El método VPT presenta algunos problemas, entre ellos que origina un número considerable de pivotes, lo que conlleva que hay que calcular muchas funciones de distancia, ya que para cada objeto en un nodo hoja, se calcula la función de distancia entre ese elemento y todos los pivotes que le anteceden desde el nodo raíz. Con un número menor de pivotes el número de distancias a calcular sería también menor. Pero al mismo tiempo no se almacenan las restantes distancias, que podrían ser

útiles para descartar objetos y evitar así el cálculo de funciones de distancia en las que se ven involucrados, en el momento de realizar una búsqueda.

Por otra parte es posible realizar una partición de una zona utilizando un pivote que no pertenece a esa zona, lo que se traduce en que un mismo pivote puede usarse para particionar zonas correspondientes a nodos que tienen el mismo nivel. Esto también implica una reducción del número de funciones de distancia a evaluar.

El método *Multi Vantage Point Tree (MVPT)* [Bozkaya y Ozsoyoglu, 1997] se originó como una alternativa que mejora los problemas mencionados. El método utiliza dos pivotes por nodo, siendo cada nodo resultado de la fusión de dos niveles del árbol originado por VPT, y dándose que todos los nodos hijo del nivel inferior fusionado usan el mismo pivote.

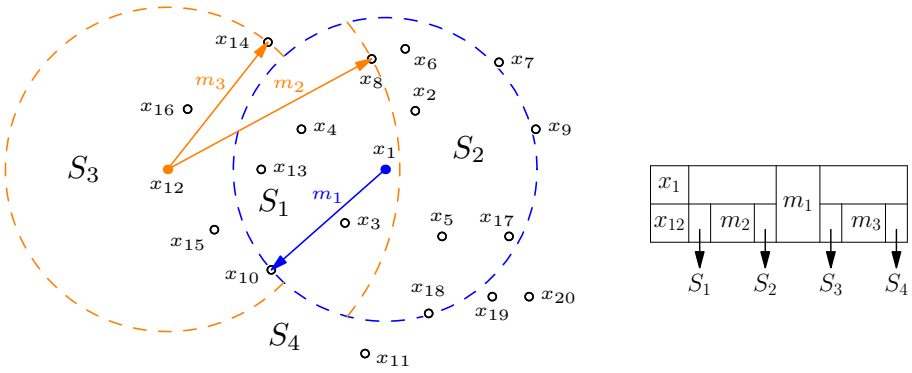


Figura 3.14: Aplicación del método MVPT.

La figura 3.14 muestra el primer nivel de particionado del método, para el mismo conjunto de objetos usado en el método VPT. El primer pivote elegido es x_1 y con la mediana de las distancias a los restantes puntos se origina una partición de dos zonas. El segundo pivote se selecciona entre los de la zona más alejada al primer pivote, y resulta ser x_{12}. Este segundo pivote crea una partición en la zona a la que pertenece, generando las regiones S_3 y S_4, pero origina también una partición en la primera zona, dando lugar a otras dos regiones S_1 y S_2. La fusión de los dos niveles del árbol hace que el nodo quede como en la figura, almacenando los dos pivotes y las distancian que definen las zonas.

Existen varias variantes del método, entre ellas una que utiliza un determinado número de percentiles en vez de la mediana, con lo que un pivote puede generar

una partición formada por más de dos zonas iniciales, dando lugar al *Multi Way Vantage Point Tree (MwVPT)* [Bozkaya y Ozsoyoglu, 1997].

3.2.7. Approximating and Eliminating Search Algorithm (AESA)

El método *Approximating and Eliminating Search Algorithm (AESA)* [Vidal, 1986, 1994][3] realiza inicialmente un preprocesado de los datos en el que calcula las distancias entre los n objetos que forman la base de datos, para poder utilizar todos los objetos como pivotes en el proceso de búsqueda, en el que se distingue una fase de aproximación o elección de los pivotes más cercanos y una fase de eliminación en la que descarta los objetos que no cumplen el criterio de búsqueda.

Reducir el número de cálculos de la función de distancia es un objetivo genérico de los métodos de búsqueda por proximidad, que se hace más necesario a medida que dicho cálculo es especialmente costoso. La estrategia que sigue AESA es utilizar un índice que contenga las distancias entre todos los objetos de la base de datos, de forma que al crearse en una etapa previa, se trasladan los costosos cálculos de las distancias a un momento anterior al proceso de búsqueda, aprovechando además esas distancias almacenadas para descartar más objetos en la fase de eliminación.

En el preprocesado las distancias entre los n objetos se almacenan inicialmente en una matriz $n \times n$ que, debido a la simetría de la función de distancia, queda reducida únicamente a $(n^2 - n)/2 = n(n-1)/2$ valores diferentes.

En la búsqueda por rango se trata de determinar si un objeto $u \in U$ cumple el criterio de búsqueda, esto es, $d(q,u) \leq r$. El proceso se inicia eligiendo un pivote p al azar y calculando su distancia al objeto de búsqueda $d(p,q)$, con lo que se determina ya si p verifica la consulta. Además por el lema 2.2 se tiene una cota inferior para la $d(q,u)$ con $u \in U$ que es:

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

El valor de la cota inferior está completamente determinado, dado que ya se ha calculado $d(q,p)$ y que se conoce $d(p,u)$ porque es una de las distancias precalculadas.

A partir de la cota inferior de la distancia se tiene el criterio de eliminación, que viene dado por

$$r < |d(q,p) - d(p,u)|$$

con lo que:

$$r < |d(q,p) - d(p,u)| \leq d(q,u)$$

[3] La diferencia fundamental entre ambos trabajos radica en el criterio de aproximación.

Dado que el criterio de eliminación es $r < |d(q,p) - d(p,u)|$, se eliminan los $u \in U$ que no cumplen que $r \geq |d(q,p) - d(p,u)|$, esto es, que no cumplen que:

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

como puede observarse en la figura 3.15, en la que los objetos situados dentro de la corona no pueden descartarse.

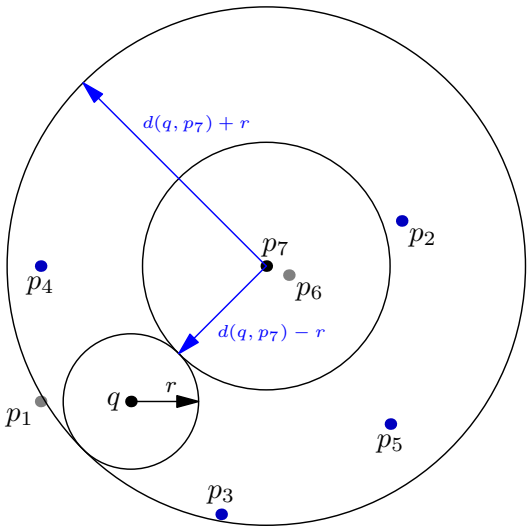


Figura 3.15: Criterio de eliminación en AESA con $p = p_7$, resultando descartados p_1 y p_6.

Entre los diferentes objetos que no resultan eliminados se reitera el proceso, pero ahora no se elige el siguiente pivote al azar, sino mediante un método de aproximación que determina el más cercano al objeto de búsqueda, usando la cota inferior dada por los objetos de los que ya se conoce la distancia a q. Se elige el más cercano y se calcula su distancia a q para poder eliminar otros objetos y así continuar el proceso.

El método AESA permite calcular también el vecino más próximo sin más que considerar en cada momento uno de los objetos como el candidato a vecino más próximo e ir sustituyéndolo por el que tenga una distancia menor.

En el caso general se considera $U_c \subset U$ el conjunto de objetos de los que se ha calculado la distancia a q. Por el lema 2.2 ocurre que:

$$\forall\, p,u \in U\,,\; q \in X \;\Rightarrow\; |d(q,p) - d(p,u)| \leq d(q,u)$$

y se denota como $d_{mci}(q,u)$ la mayor de las cotas inferiores de las distancias $d(q,u)$ para cualquier $u \in U - U_c$, que es la que se acerca más al valor real de $d(q,u)$:

$$d_{mci}(q,u) = \max_{p \in U_c}\{|d(q,p) - d(p,u)|\}$$

El criterio de eliminación para la búsqueda por rango es eliminar los $u \in U - U_c$ cuyo máximo de la cota inferior de las distancias a q es mayor que r:

$$d_{mci}(q,u) > r$$

ya que $d(q,u) \geq d_{mci}(q,u) > r$.

El criterio de eliminación para obtener el vecino más cercano es desechar los objetos $u \in U - U_c$ cuyo máximo de la cota inferior de las distancias a q es mayor que la distancia de q al candidato a vecino más cercano, u_{nn}, esto es:

$$d_{mci}(q,u) > d(q,u_{nn})$$

Con ello $d(q,u) \geq d_{mci}(q,u) > d(q,u_{nn})$ por lo que la distancia de q a u es mayor que la que ya hay al actual candidato a vecino más cercano y u puede eliminarse.

En el método AESA se considera además el conjunto $U_n \subset U$ de los objetos que, ni se ha calculado su distancia a q, ni han resultado eliminados de la lista de candidatos. Inicialmente establece $U_c = \varnothing$, $U_n = U$ y $d_{mci}(q,p) = 0, \forall p \in U$. En cada iteración, el siguiente objeto $p \in U_n$ del que se calcula su distancia a q, se elige entre los que poseen menor cota inferior de la distancia a q, esto es, cuya $d_{mci}(q,p)$ es menor. A continuación se calcula $d(q,p)$, añadiendo p al resultado si $d(q,p) \leq r$ en la búsqueda por rango, o actualizando el candidato a vecino más cercano en su caso, y entonces se eliminan los objetos de U_n que cumplen el criterio de eliminación descrito anteriormente. El proceso finaliza cuando U_n queda vacío, esto es, cuando $\forall u \in U - U_c$ $d_{mci}(q,u) > r$ en la búsqueda por rango, o $d_{mci}(q,u) > d(q,u_{nn})$ si se busca el vecino más próximo.

La figura 3.16 describe una parte del método de aproximación para un objeto p_7, concretamente el valor de las cotas inferiores determinadas sobre un conjunto de elementos $p_1, ..., p_6$. El valor mayor de esas cotas, correspondiente a p_2, es el que será la $d_{mci}(q,p_7)$, y el mínimo entre esas distancias consideradas sobre los objetos de U_n determina el siguiente objeto sobre el que hay que calcular la distancia a q.

El motivo de la utilización del menor valor de $d_{mci}(q,p)$ con $p \in U_n$ en la fase de aproximación, se debe a que se desea que p sea cercano a q porque eso incrementa el efecto de eliminación de los objetos [Vidal, 1994].

AESA es un método que concentra la complejidad en el preprocesado, donde tanto el espacio necesario como el tiempo es cuadrático, a cambio de lograr

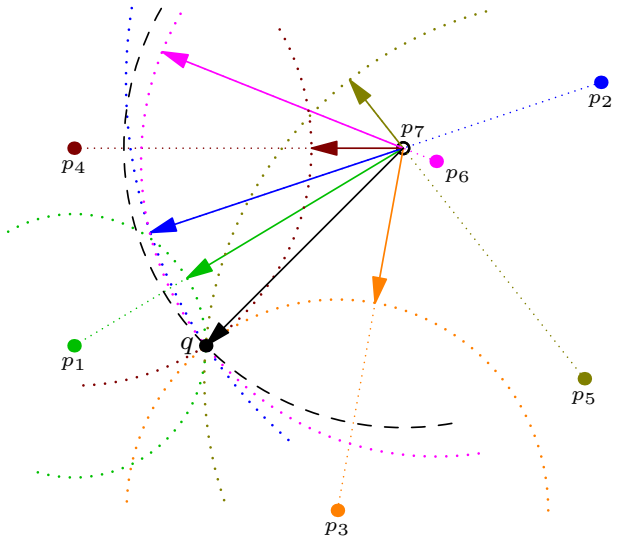


Figura 3.16: Valores de las cotas inferiores para $d(q, p_7)$ con $U_c = \{p_1, ..., p_6\}$.

sublinealidad en las búsquedas, pero aún así no resulta adecuado para grandes volúmenes de datos. Aunque su rendimiento con volúmenes reducidos de datos es muy competitivo, dado que hace uso de las distancias ya calculadas en el preprocesado, cuando se dan condiciones desfavorables, como que el radio sea un valor alto, puede acercarse a una complejidad $O(n)$, la misma que una búsqueda por fuerza bruta.

3.2.8. Lineal AESA (LAESA)

Los costosos requerimientos que presenta AESA en la fase de preprocesado, debido al cálculo y almacenamiento de las distancias entre todos los objetos de la base de datos, se han intentado reducir dando lugar al método *Lineal AESA (LAESA)* [Micó y otros, 1994], que partiendo del planteamiento de AESA, considera únicamente un número determinado de m objetos para que actúen como pivotes.

La primera consecuencia es que se pasa de calcular $n(n-1)/2$ valores diferentes a solo $n \times m$, lo que supone una reducción sustancial en la medida en que $m \ll n$, pero esto conlleva una serie de problemas; primero, solo se pueden utilizar los m objetos como pivotes, pues son los únicos de los que se conoce la distancia a los

restantes objetos de la base de datos, segundo, que el criterio de eliminación se ve afectado porque ahora podrían eliminarse pivotes que posteriormente resultasen necesarios para ajustar las cotas de las distancias, y tercero, que se plantea un nuevo problema que es la elección efectiva de los pivotes, ya que tanto su número como su localización, entre ellos y con respecto a los demás objetos, afectan sensiblemente al rendimiento del método.

La realización de las búsquedas en LAESA es similar a como se efectúa en AESA. Al igual que en este último, se considera $U_c \subset U$ el conjunto de objetos de los que se ha calculado la distancia a q y el conjunto $U_n \subset U$ de los objetos que ni se ha calculado su distancia a q ni han resultado eliminados de la lista de candidatos. El problema estriba ahora en que los objetos que pertenecen a U_c y que no son pivotes, al no disponer de sus distancias a los objetos de U_n no permiten eliminar elementos de U_n. Los autores proponen varias estrategias para resolver la situación, como elegir siempre un pivote antes que un objeto no pivote en U_n y eliminar pivotes de U_n solo si un porcentaje de pivotes está ya en U_c.

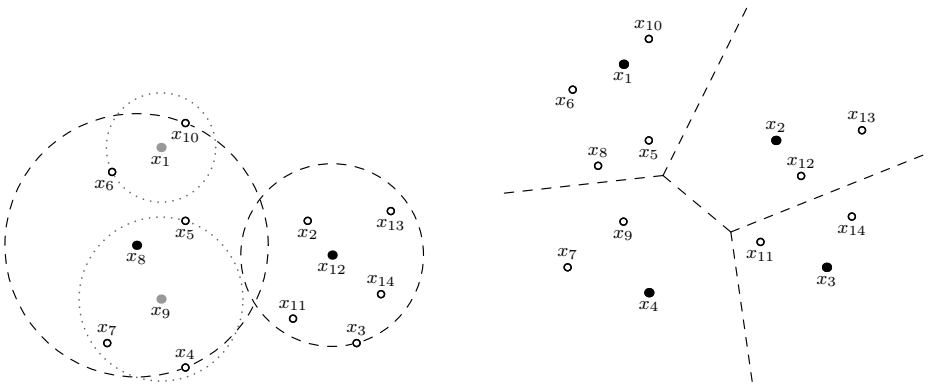
En relación a la elección de los m objetos que serán pivotes, los autores proponen un algoritmo que escoge los objetos que están tan lejos unos de otros como sea posible, esto es, que presentan unas distancias máximas entre ellos.

Tanto AESA como LAESA son métodos estáticos, necesitando que el índice se construya disponiendo de la totalidad de los objetos. Las operaciones de inserción y borrado pueden degradar el rendimiento debido a los cálculos necesarios para actualizar el índice.

3.3. Métodos basados en particiones

Los métodos basados en particiones dividen el espacio en un conjunto de zonas formando una partición, como ya se expuso en la página 38 de la sección 2.4.2. Consideran un objeto de referencia o centro para cada zona y establecen un criterio de pertenencia de los objetos a las zonas basado en su proximidad al centro, bien estableciendo el radio de cobertura de cada zona, o distancia máxima del centro a un elemento de la zona, lo que origina una partición mediante zonas esféricas (lado izquierdo de la figura 3.17), bien determinando que un objeto pertenece a una zona si está a una distancia menor del centro de esa zona que de cualquier otro centro, lo que implica que las zonas se delimitan mediante un conjunto de hiperplanos (lado derecho de la figura 3.17), siendo cada uno de ellos la frontera entre dos zonas contiguas. Utilizan un índice para almacenar datos relevantes relativos a los centros, con el objetivo de que, ante una búsqueda determinada y después de calcular las distancias del objeto de búsqueda a los centros, se puedan descartar zonas enteras

sin realizar el cálculo de las distancias del objeto de búsqueda a los elementos de las zonas.



(a) Zonas esféricas (b) Hiperplanos

Figura 3.17: Métodos de particionado.

A continuación se describen los métodos más representativos basados en particiones.

3.3.1. Bisector Tree (BST)

Uno de los primeros métodos de búsqueda por similitud basados en particiones es el conocido como *Bisector Tree (BST)* [Kalantari y McDonald, 1983]. Usa una estructura arbórea para representar la partición del espacio y almacena el radio de cobertura de cada zona para usarlo como criterio de eliminación de zonas en el proceso de búsqueda.

El proceso de construcción consiste inicialmente en particionar el espacio en dos zonas, eligiendo en cada una de ellas un objeto que actúa como centro, y asignar a cada zona los objetos que están más próximos a su correspondiente centro. El proceso se repite recursivamente sobre los objetos pertenecientes a cada zona, reflejando en una estructura arbórea el resultado del particionado, colocando en cada nodo los dos centros, por ejemplo c_1 y c_2, y creando un subárbol para los elementos que pertenecen a cada zona, por ejemplo los elementos más cercanos a c_1 se asignan al subárbol de la izquierda y los más cercanos a c_2 se asignan al subárbol de la derecha.

La figura 3.18 representa los diferentes niveles del particionado de un conjunto de elementos de $\mathbb{R}^2$ y la 3.19 el índice resultante. Se consideran más objetos que en la figura 3.17 para describir mejor el particionado.

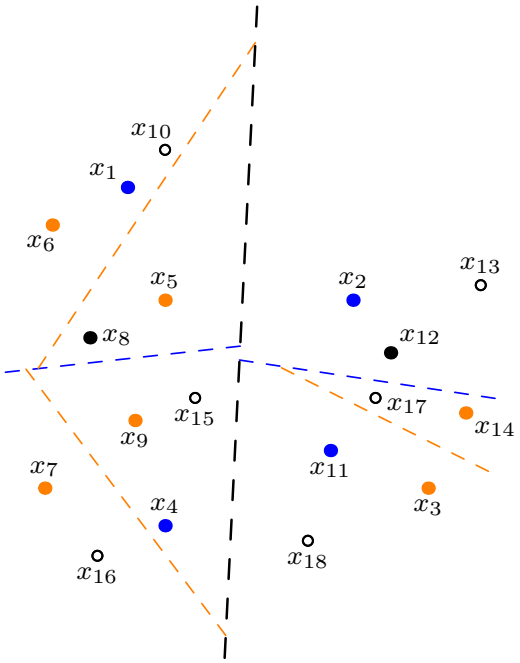


Figura 3.18: Particionado mediante hiperplanos en el método BST.

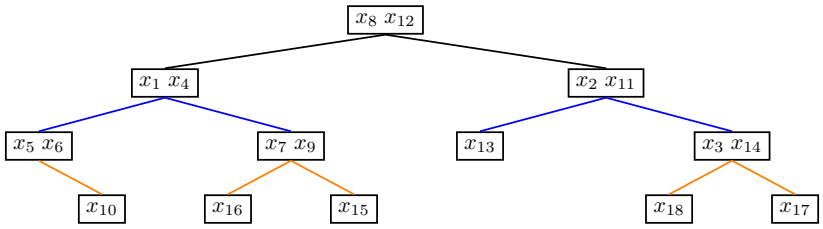


Figura 3.19: Índice del método BST.

Una particularidad de BST es que también almacena para cada centro el radio de cobertura o mayor distancia de cada centro a otro objeto de su misma zona.

Se tienen así dos centros de zonas que determinan una partición y forman un nodo del índice, cada zona contiene a un conjunto de objetos que formarán los subárboles asociados al nodo y existe también una región esférica para cada zona, determinada por el centro de la zona y su radio de cobertura. La intersección de las regiones determinadas por los radios de cobertura puede no ser vacía, a diferencia de lo que ocurre con las zonas. El proceso de particionado se aplica recursivamente en cada nodo.

Dada una búsqueda por rango $R(q, r)$ se recorre el índice comparando en cada nodo los centros con el objeto de búsqueda y se aplica el criterio para descartar una zona C_i basado en el radio de cobertura, que, como se vio en la ecuación 2.13 de la página 43, es consecuencia del lema 2.4 y viene dado por:

$$d(q, c_i) - r_{c_i} > r \qquad \text{o} \qquad d(q, c_i) - r > r_{c_i}$$

para un centro c_i y un radio de cobertura r_{c_i}. Esto significa que una zona puede descartarse si el círculo de búsqueda no interseca a la región determinada por el centro y el radio de cobertura. Podría ocurrir que no se descartase ninguna zona.

La figura 3.20 muestra un ejemplo en el que los puntos x_8 y x_{12} se eligen como centros de las dos zonas delimitadas por el segmento discontinuo. La parte inferior refleja el estado final del índice. Para realizar la búsqueda se parte de la raíz del índice y se sigue el criterio expuesto. Dado que $d(q, x_8) - r > r_{x_8}$ se prescinde del subárbol de la izquierda, lo que implica que la zona de la izquierda se descarta. En cambio, el subárbol de la derecha no puede descartarse porque $d(q, x_{12}) - r < r_{x_{12}}$, con lo que se recorrería el subárbol de la derecha y se aplicaría de nuevo el proceso de búsqueda usando ahora el radio de cobertura sobre los dos centros x_2 y x_{11}.

3.3.2. Voronoi Tree (VT)

Una modificación de BST que mejora el rendimiento de las búsquedas se denomina *Voronoi Tree (VT)* [Dehne y Noltemeier, 1987]. Las diferencias con BST consisten en utilizar dos o tres centros en cada nodo interno, y en que cuando se crea un nodo para almacenar un nuevo elemento, su objeto más cercano del nodo padre se introduce también en el nuevo nodo. El radio de cobertura se va reduciendo a medida que se desciende en el árbol, originando que las zonas sean más pequeñas y permitiendo que en la búsqueda se descarten mejor. Inicialmente se presenta como un método que proporciona un mejor rendimiento en las búsquedas que BST. Posteriormente se demostró [Noltemeier, 1989] que con VT se pueden obtener árboles balanceados con un procedimiento de inserción similar al de los B-Trees.

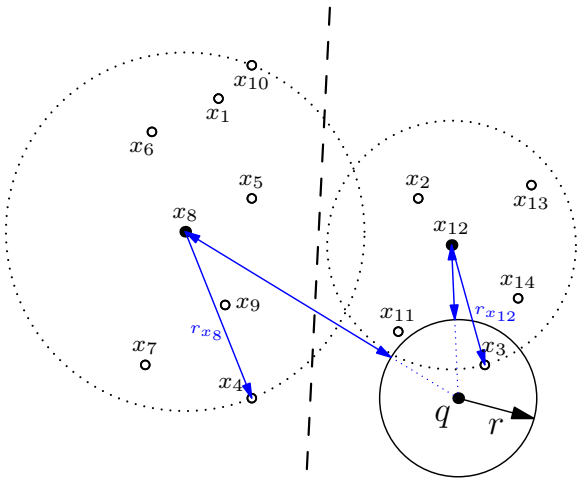


Figura 3.20: Fase inicial del proceso de búsqueda en el método BST.

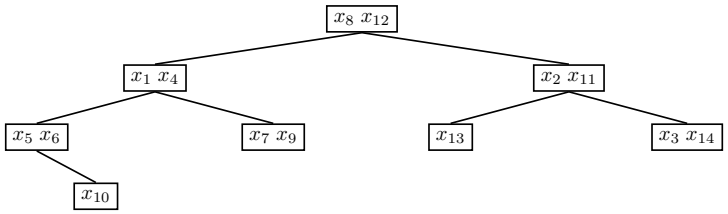


Figura 3.21: Índice completo del método BST para el conjunto de objetos sobre los que se busca.

3.3.3. Generalized Hyperplane Tree (GHT)

El método denominado *Generalized Hyperplane Tree (GHT)* [Uhlmann, 1991] es idéntico al BST en el proceso de construcción del particionado y del índice asociado, excepto que no almacena los radios de cobertura y que, por tanto, difiere en la regla para descartar las zonas en tiempo de búsqueda, ya que utiliza el criterio del hiperplano, consecuencia del lema 2.5 y expresado en la ecuación 2.16 de la página 47.

Aplicando el criterio para los centros c_1 y c_2, no se puede descartar el subárbol de la izquierda correspondiente a c_1 y, por tanto, hay que recorrerlo, si:

$$d(q, c_1) - r \ \leq \ d(q, c_2) + r \qquad \text{con } d(q, c_2) \ \leq \ d(q, c_1)$$

y análogamente, se recorre el subárbol de la derecha si:

$$d(q, c_2) - r \;\leq\; d(q, c_1) + r \qquad \text{con } d(q, c_1) \;\leq\; d(q, c_2)$$

Al igual que en el BST, puede ocurrir que no se descarte ningún subárbol. Como no utiliza los radios de cobertura, la construcción del índice evita un número importante de evaluaciones de la función de distancia.

La figura 3.22 representa la aplicación del proceso de búsqueda del método GHT, en su primera fase, al mismo conjunto de objetos que en el ejemplo del BST, que se ha particionado también de la misma manera, por lo que la estructura del índice es la misma que la de la figura 3.21, siendo los dos primeros centros x_8 y x_{12}.

Para el subárbol de la izquierda, correspondiente al centro x_8, se aprecia que:

$$d(q, x_8) - r \;>\; d(q, x_{12}) + r$$

porque se ha desplazado la distancia correspondiente al segundo miembro de la desigualdad, aplicando que $d(q, x_{12}) + r \;=\; d(q, x_{12}) - r + 2r$, con lo que puede descartarse ese subárbol y por tanto la zona de centro x_8.

Para el subárbol de la derecha ya no se puede aplicar el criterio de descarte, porque solo se puede aplicar a la zona cuyo centro está más alejado de q, por lo que la zona de centro x_{12} no puede descartarse y entonces hay que recorrer el subárbol de la derecha y reiterar el criterio sobre los centros situados en el siguiente nodo, que son x_2 y x_{11}.

3.3.4. Geometric Near-neighbor Access Tree (GNAT)

Una propuesta que generaliza el árbol binario del método GHT a un índice arbóreo de grado m, es la conocida como *Geometric Near-neighbor Access Tree (GNAT)* [Brin, 1995].

En el primer nivel se eligen m centros $\{c_1, \ldots, c_m\}$ que originan una partición del espacio en m zonas $\{C_1, \ldots, C_m\}$, formadas por los elementos más cercanos a cada centro, esto es:

$$C_i \;=\; \{x \in U \;/\; \forall j \neq i \quad d(x, c_i) < d(x, c_j)\}$$

Partiendo del nodo raíz que contiene los m centros, se originan m nodos hijos con los elementos correspondientes a cada zona C_i. Además en cada nodo se almacena una matriz $m \times m$ con las distancias mínimas y máximas de cada centro a cada zona, esto es:

$$r_{ij} \;=\; [\; \min_{x \in C_j}(x, c_i)\,,\; \max_{x \in C_j}(x, c_i)\;]$$

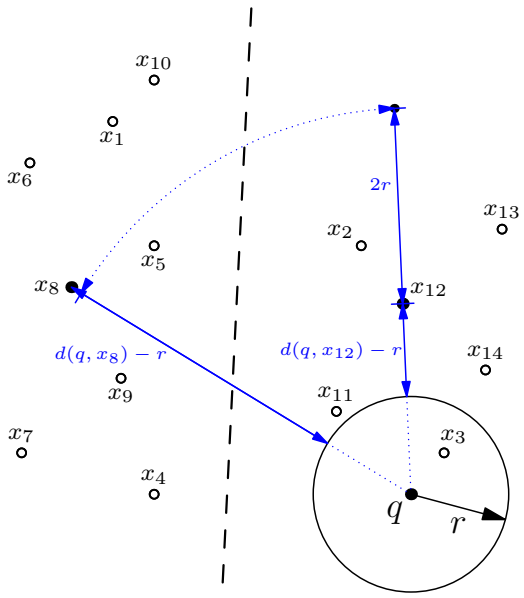


Figura 3.22: Fase inicial del proceso de búsqueda en el método GHT.

En el proceso de realización de una búsqueda $R(q, r)$, el objeto de búsqueda q se compara con uno de los centros c_i de la raíz, obteniendo $d(q, c_i)$, lo que permite descartar las zonas C_j correspondientes a otros centros c_j si $d(q, c_i) \pm r$ no interseca a r_{ij}. El proceso se repite con otros centros hasta que ya no se puede descartar a ninguno más. En ese caso se repite el procedimiento recursivamente sobre cada subárbol no eliminado. A lo largo del proceso se van detectando también los centros que cumplen el criterio de búsqueda.

En la figura 3.23 se muestra una partición con centros x_1, x_2, x_3, x_4. En la subfigura superior se utiliza inicialmente el centro x_3 para intentar descartar las zonas. Los valores mínimo y máximo de la matriz aparecen en trazos punteados a color. Primero se obtiene que la zona del propio x_3 no resulta descartada, porque el círculo de búsqueda interseca a $r_{33} = [0, d(x_3, x_{11})]$. No se representó la zona correspondiente a x_2 porque al estar el círculo de búsqueda en C_2, no puede descartarse. La zona de x_4 tampoco puede descartarse, al generar una pequeña intersección con el círculo de búsqueda. La única zona que puede descartarse con x_3 es la de x_1 porque la región delimitada por los valores de la matriz, que aparece parcialmente con trazo punteado en naranja, no corta al círculo de búsqueda.

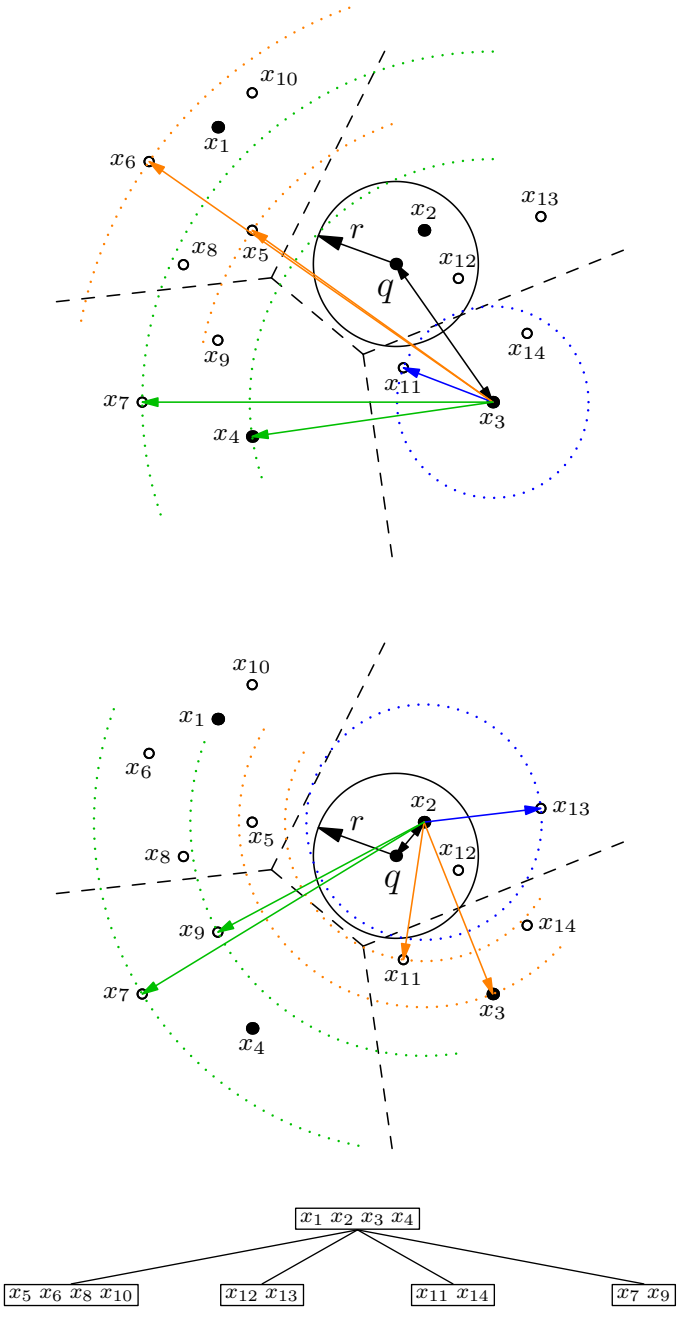


Figura 3.23: Método GNAT.

En la subfigura central se repite el proceso usando el centro x_2. Primero el cálculo de $d(q, x_2)$ indica que el centro cumple el criterio de búsqueda. Se aprecia que las restantes zonas no descartadas con el centro anterior, correspondientes a x_3 y x_4, resultan ahora descartadas utilizando x_2. Al no poder descartarse más zonas, esto finaliza la búsqueda en el primer nivel del índice, en el que ya solo se explora el subárbol correspondiente a x_2.

3.3.5. M-Tree (MT)

Uno de los métodos más importantes en la indexación y búsqueda en espacios métricos es el *M-Tree (MT)* [Ciaccia y otros, 1997]. Además de proporcionar capacidades dinámicas que permiten la inserción y el borrado de elementos, partiendo incluso de una base de datos vacía, presenta también un buen rendimiento en las búsquedas y un reducido número de evaluaciones de la función de distancia.

El procedimiento de construcción de la estructura es similar al utilizado en GNAT, ya que se parte de una serie de centros por nodo, de forma que los elementos más cercanos a cada centro originan un subárbol en el índice, representando una partición del espacio.

El procedimiento de búsqueda es diferente a GNAT y resulta más parecido a BST, porque MT almacena también el radio de cobertura de cada zona y al producirse una consulta, el objeto de consulta se compara con todos los centros y se aplica el criterio del radio de cobertura para descartar las zonas.

La característica principal de MT es la manera de gestionar la introducción de nuevos elementos, a la vez se mantienen unas buenas condiciones de rendimiento. Un nuevo elemento se coloca en el que se considera el mejor subárbol, entendiendo por ello no necesariamente el correspondiente al centro más cercano, sino el que origina un menor incremento del radio de cobertura una vez colocado el nuevo elemento en él.[4] Si se dan empates, entonces se elige el centro más cercano. El proceso de inserción se realiza recursivamente hasta llegar a que el nuevo elemento se añade al nodo hoja correspondiente, almacenando también la distancia al centro de su nodo padre. Si ocurre que se desborda el nodo hoja, entonces se divide en dos, haciendo que se elijan dos elementos del nodo como centros de los nuevos nodos, y otro de los elementos se promociona al nodo padre, de la misma forma que se hace en los B-Trees, lo que conlleva que el árbol se mantiene balanceado.

[4] Como ya se comentó en la exposición del método VT, cuanto menor es el radio de cobertura, mayores son las posibilidades de descartar más objetos.

3.3.6. Spatial Approximation Tree (SAT)

El diagrama de Voronoi sirve de inspiración al método *Spatial Approximation Tree (SAT)* [Navarro, 1999], un método inicialmente estático al que luego se le han incorporado características dinámicas [Navarro y Reyes, 2002].

Partiendo de una serie de puntos, el diagrama de Voronoi representa las zonas o celdas correspondientes a cada punto, que están más cercanas a ese punto que a los restantes. Asociado al diagrama se tiene el grafo o triangulación de Delaunay[5], en el que los nodos son los puntos y hay un arco entre dos nodos si en el diagrama sus celdas tienen una frontera común, esto es, el grafo representa la relación de vecindad existente en el diagrama. La construcción del grafo sería un problema NP-completo, por lo que el método SAT intenta obtener una aproximación al grafo con una estructura arbórea que aprovecha la relación de proximidad entre los puntos y que tenga las suficientes conexiones para permitir las búsquedas, sin necesidad de calcular un número elevado de funciones de distancia.

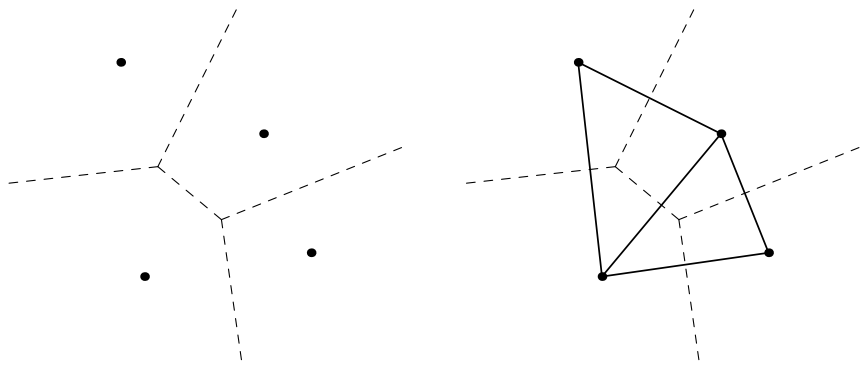


Figura 3.24: Diagrama de Voronoi y su grafo de Delaunay.

Inicialmente se parte de un punto tomado aleatoriamente, que será el nodo raíz del árbol, que denotamos como a, y al que se le asocia el conjunto más pequeño $N(a)$, denominado vecinos de a, cuyos elementos tienen la propiedad de estar más cerca de a que de cualquier otro elemento de $N(a)$.

[5]Boris Nikolaevich Delaunay fue un matemático ruso nacido en 1890 y que vivió noventa años. Cuando era niño sus padres veraneaban en los Alpes, lo que le hizo aficionarse a la escalada y llegó a ser uno de los escaladores soviéticos más importantes. También aprendió el alemán y el francés, de hecho su apellido es la versión francesa que utilizaba en sus publicaciones en ese idioma. Su triangulación tiene la propiedad de que las circunferencias circunscritas en cada triángulo no contienen a ningún vértice de otro triángulo.

La definición de $N(a)$ no genera un conjunto único y dado que se hace en función de sí mismo, la búsqueda del conjunto mínimo originaría un problema NP-completo. Por ello se recurre a una heurística [Samet, 2006] que considera los puntos en el orden determinado por su distancia a a, y los incorpora en ese orden a $N(a)$ si cumplen la condición de proximidad dada. En el nodo se almacena también la máxima distancia de a a los elementos de $N(a)$.

El proceso de construcción se aplica recursivamente a cada elemento en $N(a)$ pasando cada uno a ser un nodo hijo de a sobre el que se calculará su conjunto de vecinos.

Al realizar una búsqueda $R(q, r)$, el método de descarte propuesto en SAT se basa en la ecuación 2.16 de la página 47, consecuencia del lema 2.5. Siendo c el vecino más cercano a q entre un conjunto de vecinos N, hay que explorar los nodos correspondientes a cualquier $b \in N$ tal que $d(q, b) - r \leq d(q, c) + r$.

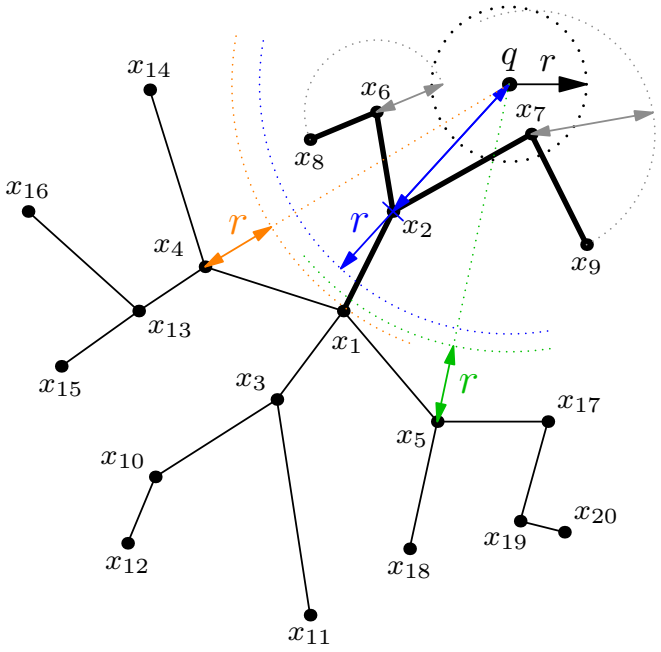


Figura 3.25: Método SAT.

En la figura 3.25 se representa un ejemplo en el que la estructura comienza en x_1 y el primer conjunto de vecinos es $N(x_1) = \{x_2, x_3, x_4, x_5\}$. Por ejemplo x_8

no forma parte de $N(x_1)$ porque al entrar x_2 en $N(x_1)$, resulta que x_8 está más próximo a otro elemento de $N(x_1)$, que es x_2, que a x_1.

Cada uno de los elementos de $N(x_1)$ se convierte por el proceso recursivo, en la raíz de un subárbol que llevará a los elementos que resulten sus vecinos, como por ejemplo $N(x_4) = \{x_{13}, x_{14}\}$.

En la gráfica se ha resaltado con un trazo grueso el recorrido necesario para resolver la consulta. Para los subárboles correspondientes a x_4 y a x_5 se muestra que no es necesario recorrerlos porque cumplen el criterio de descarte usando $c = x_2$. Lo mismo ocurre con x_3, incluso de forma más obvia. Finalmente hay que comprobar la distancia a q tanto para x_8 como para x_9.

3.3.7. Lista de Clusters (LC)

La mayoría de los métodos basados en particiones dividen el espacio de manera recursiva, empezando con dos o más zonas que a su vez vuelven a dividirse. De esta manera se va creando un índice arbóreo que refleja la descomposición del espacio y que sirve de soporte para realizar las operaciones de búsqueda.

El método *Lista de Clusters (LC)* [Chávez y Navarro, 2000] plantea una alternativa en la que no se produce un particionado recursivo, sino que se genera una partición formada por una serie de zonas, determinadas en función de sus centros y de sus radios de cobertura, que ya no sufrirán ninguna división posterior. La elección de los centros resulta determinante, ya que origina una lista ordenada de zonas, que es la estructura que da nombre al método, y sobre la que se realizarán posteriormente las búsquedas, permitiendo, en algún caso, descartar todas las zonas que siguen a una determinada en la lista.

El proceso de construcción se inicia eligiendo el primer objeto como centro de la primera zona y estableciendo un límite para dicha zona, bien en función del número máximo de objetos que puede contener, bien determinando un radio de cobertura máximo. Una vez que se ha completado la primera zona, se repite el procedimiento con el resto de objetos de la colección. Se tiene así un primer centro c_1 y su radio de cobertura r_{c_1}, lo que origina la primera zona:

$$C_1 = (c_1, r_{c_1}) = \{\, x \in U \,/\, d(x, c_1) \leq r_{c_1} \,\}$$

Al finalizar se tiene una lista ordenada de zonas $C_1 \to C_2 \to \cdots \to C_n$.

Tanto la selección de los centros como de los radios de cobertura puede realizarse de varias maneras. En el caso de los centros los autores han propuesto elegir como siguiente centro, el objeto que maximiza la suma de las distancias a los centros ya seleccionados. Con respecto al radio de cobertura, podría usarse un valor fijo para

todas las zonas o el que resultase para que cada zona tuviese el mismo número de elementos; en estudios experimentales los autores muestran que esta última opción aporta un mayor rendimiento [Chávez y Navarro, 2005]. El método obtiene resultados muy competitivos en dimensiones altas.

Ante una búsqueda $R(q,r)$ se calcula $d(q,c_1)$ para, además de determinar si c_1 cumple o no la condición de búsqueda, usarlo en el criterio para descartar o no las zonas. Según la ecuación 2.13 y como consecuencia del lema 2.4, la zona C_1 no se podría descartar, y por tanto habría que explorarla, si:

$$d(q,c_1) - r_{c_1} \leq r$$

El método permite además determinar si se procesa o no el resto de zonas de la lista, teniendo que hacerlo si:

$$d(q,c_1) + r > r_{c_1}$$

debido a que si se produjera que $d(q,c_1) + r \leq r_{c_1}$ entonces ocurriría que la zona de centro c_1 contendría al círculo de búsqueda y ya no sería necesario explorar las zonas restantes.

El proceso se repite para cada zona de la lista, hasta que se puedan descartar todas las zonas a partir de una dada, o hasta que se llegue a la última de las zonas.

La figura 3.26 representa una base de datos de veinte puntos que se ha particionado en tres zonas siguiendo el método LC y utilizando como centros $\{x_{13}, x_5, x_6\}$. El orden seguido en la selección de los centros es primero x_{13}, que determina la zona de color azul y que tiene como radio de cobertura la distancia entre el centro y x_{12}; la segunda zona tiene como centro a x_5, como radio de cobertura la distancia del centro a x_{11} y a ella no pertenecen ni x_1 ni x_3, por estar ya en la de x_{13}; y finalmente la última zona tiene de centro a x_6 y su radio de cobertura es la distancia del centro a x_7. La lista de clusters quedaría de la forma:

$$(x_{13}, r_{x_{13}}) \rightarrow (x_5, r_{x_5}) \rightarrow (x_6, r_{x_6})$$

Para realizar la consulta $R(q_1,r)$ se comienza por la zona del primer centro x_{13}, calculando $d(q_1,x_{13})$. La zona puede descartarse porque $d(q_1,x_{13}) - r_{x_{13}} > r$. Se comprueba si pueden descartarse las restantes zonas de la lista, resultando que no porque $d(q_1,x_{13}) + r > r_{x_{13}}$.

Se procede con la segunda zona, la correspondiente a x_5, en la que se obtienen los mismos resultados: la zona se descarta pero no pueden eliminarse las restantes zonas de la lista.

En la tercera zona ocurre que $d(q_1,x_6) - r_{x_6} < r$ con lo que no puede descartarse y hay que calcular las distancias de q_1 a los dos puntos restantes x_7 y x_8, resultando que solo x_7 cumple el criterio de búsqueda.

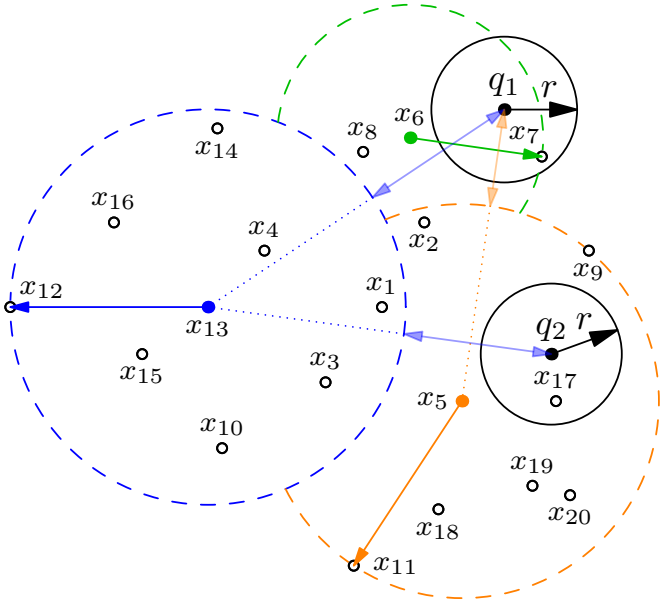


Figura 3.26: Método LC.

La consulta $R(q_2, r)$ permite plasmar el criterio de descarte de las restantes zonas de la lista. Primero se comienza por x_{13} obteniendo que puede descartarse esa zona, pero no las restantes. Al actuar sobre la zona de centro x_5 resulta que esta debe explorarse; calculando las distancias de q_2 a sus puntos se determina que x_{17} verifica el criterio de búsqueda. Ahora se comprueba si pueden descartarse las restantes zonas, para lo que se halla $d(q_2, x_5) + r$ que es menor que el radio de cobertura r_{x_5} por lo que pueden descartarse las zonas siguientes en la lista, en este caso, solo la última de centro x_6 .

3.4. Otros métodos

Los métodos analizados hasta este punto están entre los más destacados en la búsqueda por similitud sobre espacios métricos, tanto por la aplicación de determinadas ideas, en su momento innovadoras, y que van desde la estructura del índice a la heurística aplicada, como por ser ejemplos de buen rendimiento en la realización de las búsquedas, ya que en algunos casos, a pesar de que su desarrollo

data ya de hace unos años, siguen siendo métodos de referencia para evaluar el rendimiento de nuevas alternativas.

Sin embargo, sobre todo a lo largo de la última década, se han desarrollado otros métodos. Algunos de ellos llevan a cabo aportaciones interesantes por lo que resulta adecuado realizar una descripción de sus características.

Sparse Spatial Selection (SSS)

El método *Sparse Spatial Selection (SSS)* [Brisaboa y otros, 2006] es una técnica de selección de pivotes que asegura la elección de un conjunto de pivotes bien distribuidos en el espacio, en el sentido de que cada objeto de la base de datos tendrá un pivote cercano.

La elección de los pivotes se realiza a medida que los objetos se incorporan a la base de datos. El primer objeto se elige como pivote. A medida que se consideran otros objetos, cada uno de ellos será, o un nuevo pivote, o un objeto más de la colección. Se determina si un objeto es o no un pivote, en función de que su distancia al resto de pivotes actuales sea mayor que $M\alpha$, siendo M la distancia máxima entre dos objetos cualesquiera del espacio, y α un parámetro relacionado con la densidad de los pivotes en el espacio, que toma valores en el intervalo $(0, 1]$.

El proceso de manera detallada comienza determinando los valores de M y α. El primer objeto de la base de datos se elige como pivote. Al considerar el siguiente objeto, el proceso se convierte en iterativo actuando de la forma:

1. Se compara el nuevo objeto x con cada uno de los pivotes p_i siendo $i = \{1, \ldots, m\}$.
2. Si $d(x, p_i) \geq M\alpha$ para todo i con $i = \{1, \ldots, m\}$, entonces x pasa a ser un nuevo pivote. Se calculan las distancias del nuevo pivote a cada uno de los objetos de la base de datos y se almacenan en el índice.
3. Si $d(x, p_i) < M\alpha$ para algún i con $i = \{1, \ldots, m\}$, entonces x es un objeto más de la colección. Se calculan las distancias de x a los pivotes actuales y se almacenan en el índice.

Dependiendo del valor de α se tendrán más o menos pivotes. Los autores han comprobado experimentalmente que el número óptimo de pivotes se da si α está entre $0,35$ y $0,4$. Para valores mayores de α se tiene un número menor de pivotes, con lo que su densidad será también menor y las distancias de los pivotes a los objetos, serán, en general, mayores. Por el contrario, si α toma valores menores que los mencionados, la densidad de los pivotes será mayor, y, en general, los objetos dispondrán de pivotes que estarán más cerca de ellos.

Una característica importante de SSS es que determina un conjunto de pivotes óptimo con un coste mínimo, porque utiliza las distancias ya calculadas al incorporar los objetos a la base de datos.

Las búsquedas en SSS se realizan de la misma manera que en otros métodos basados en pivotes.

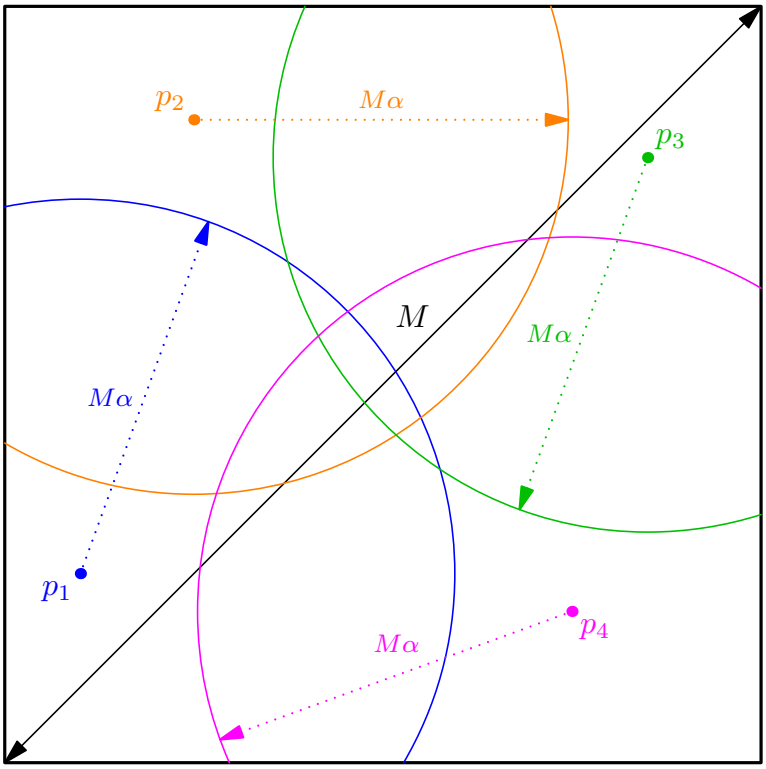


Figura 3.27: Selección de pivotes mediante SSS con $\alpha = 0,35$.

La figura 3.27 muestra un ejemplo de un espacio en el que SSS ha seleccionado cuatro pivotes. El valor de M es la máxima distancia posible entre objetos, determinada en este caso por el segmento existente entre dos vértices opuestos. Se ha considerado que $\alpha = 0{,}35$. El primer objeto se ha elegido como el primer pivote, p_1. Los siguientes objetos se van incorporando hasta que uno de ellos está a una distancia de p_1 mayor que $M\alpha$, pasando a ser el siguiente pivote, p_2. Así

se van eligiendo los pivotes hasta que con cuatro pivotes se recubre todo el espacio. A partir de este momento ya no se seleccionarán más pivotes.

Si se eligiera un valor de α menor que $0,35$, el valor de $M\alpha$ sería también menor y esto ocasionaría que la densidad de los pivotes fuese mayor. Esta situación se observa en la figura 3.28 en la que se ha considerado un valor de $\alpha = 0,25$, con lo que se necesita un pivote más para recubrir el mismo espacio que antes.

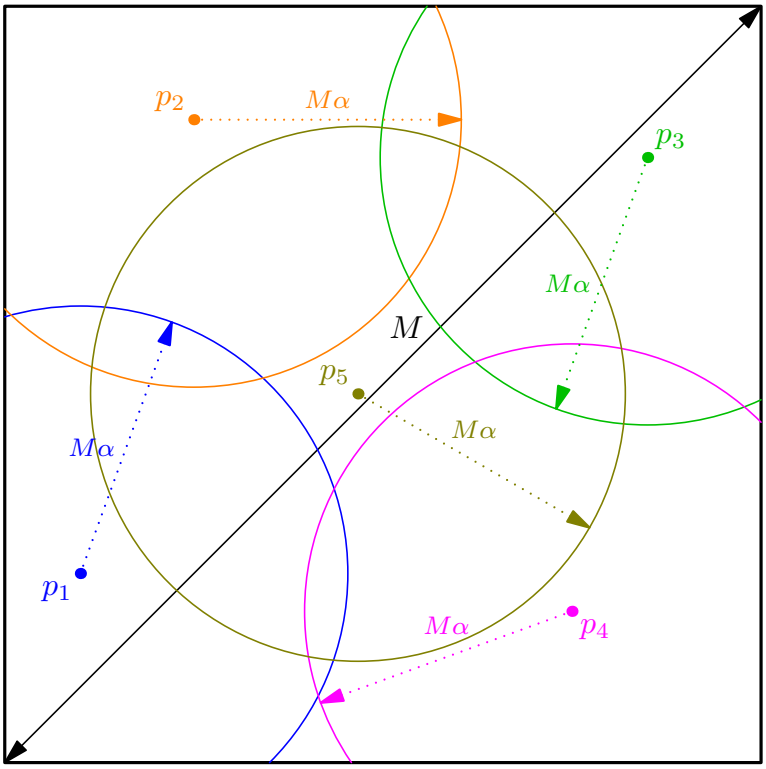


Figura 3.28: Selección de pivotes mediante SSS con $\alpha = 0,25$.

Que el método dependa del parámetro α, que determina la densidad de los pivotes en el espacio, permite utilizar SSS para elegir pivotes que resulten más cercanos o más lejanos a los objetos de la base de datos, según interese. Considerando un α menor, el método garantiza pivotes más cercanos a los objetos y con ello un número mayor de pivotes, como ocurre en la figura 3.28, a diferencia de la 3.27.

Esta característica de SSS es destacable, porque supone centrarse en un aspecto a menudo no muy estudiado, como es la selección efectiva de qué pivotes utilizar y cuántos considerar. Este método proporciona un mecanismo para realizar la selección de un conjunto óptimo de pivotes distribuidos uniformemente en el espacio, pero también permite modificar su densidad. Esta cualidad del método se utilizará posteriormente en el capítulo 4, para garantizar una seleción de pivotes que se encuentren realmente cerca de los objetos de la base de datos.

Los pivotes elegidos pueden variar incluso para la misma colección, dependiendo, por ejemplo, del orden en el que se consideran los objetos, pero aún así, el rendimiento de diferentes conjuntos de pivotes seleccionados para la misma colección es prácticamente similar, lo que refuerza la idea de que las características del conjunto de pivotes seleccionados dependen realmente del propio espacio y no estrictamente del conjunto de pivotes.

Dos características importantes del método son que es adaptativo y también dinámico. Que sea adaptativo es que por sí mismo determina el número óptimo de pivotes del espacio, según las características de la colección y de manera independiente del tamaño de la base de datos. Que sea dinámico implica que permite las operaciones de borrado, inserción y actualización en la colección de objetos. Incluso puede aplicarse a una colección inicialmente vacía y a la que se le incorporan objetos de manera progresiva.

Se ha tratado el método SSS en profundidad porque se utilizará más adelante en esta tesis. A continuación se describen de manera muy breve otros métodos.

Variantes de M-Tree

El método de M-Tree ha dado origen a muchas variantes. Se comentan algunas de ellas.

El método *Slim-Tree* [Traina Jr. y otros, 2000, 2002] es una variación del método M-Tree que acelera la inserción en los nodos y la división de los mismos. Aplica un algoritmo denominado *Slim-down* que intenta reducir el solapamiento entre las regiones asociadas a los nodos, trasladando las entradas entre nodos contiguos. La consecuencia es que reduce el número de accesos a disco, en relación a los que se obtenía con M-Tree.

El método *Pivoting M-Tree (PM-Tree)* [Skopal y otros, 2004] es otra variante del M-Tree. Consiste en incorporar a M-Tree un conjunto de pivotes que permiten dividir los clusters iniciales según las regiones determinadas por esos pivotes, a las que denominan hiperanillos, con lo que se logra un mejor ajuste del tamaño de los clusters en la realización de las búsquedas.

M-Index

El método *Metric Index (M-Index)* [Novak y Batko, 2009; Novak y otros, 2011] elige un conjunto de objetos de referencia, o centros de clusters, para realizar un particionado mediante hiperplanos. El particionado es recursivo para su aplicación a grandes volúmenes de datos. Utiliza una estructura arbórea para el índice.

Se caracteriza porque establece una correspondencia entre el espacio métrico y un dominio de valores reales, de forma que a cada objeto de la colección le hace corresponder un valor real que mantiene el criterio de proximidad de los datos, esto es, que a objetos similares les hace corresponder valores cercanos. El valor real asignado a un objeto consta de una parte entera, que identifica al cluster, y de una parte decimal, que es la distancia al centro del cluster más cercano.

3.5. Acerca del análisis de las soluciones

Los espacios métricos constituyen un marco teórico que permite formalizar la búsqueda por similitud, pero los requerimientos formales que utilizan son muy reducidos, imponiendo solo unas propiedades relativamente sencillas, lo que permite encontrar diferentes funciones de distancia que las cumplen. Esto ha permitido que el marco de trabajo sea muy amplio, pero precisamente por ello, en ocasiones se pierden características relevantes de los espacios métricos, ligadas a veces al dominio de aplicación, y otras veces a las particularidades de las colecciones, como por ejemplo, la distribución de los objetos en el espacio.

Estas características específicas de los espacios métricos han supuesto un cierto problema para el estudio analítico de los métodos. Por una parte, se considera que son demasiado importantes para ignorarlas, pero a la vez, resultan difíciles de modelar. A menudo los análisis formales realizados no se han visto confirmados por los resultados experimentales, algo que resulta excepcional, lo que ha llevado a que, o se recurra a la experimentación, o se obtengan complejos modelos de coste. Diversos trabajos han reflejado estas circunstancias [Ciaccia y otros, 1998; Navarro, 2009; Bustos y Morales, 2010]. Los análisis algorítmicos han determinado que, en general, tanto los métodos basados en pivotes como los basados en particiones presentan una complejidad $O(n^{\alpha})$ con $0 \leq \alpha \leq 1$.

Actualmente la comunidad científica acepta que el análisis asintótico no es útil para comprobar el rendimiento de un método, ni para comparar unos métodos con otros [Navarro, 2009], aunque puede ser una buena herramienta para entender las causas que determinan el comportamiento de los métodos y servir también como guía para el desarrollo de nuevos algoritmos. Por este motivo la comprobación del rendimiento de un método debe basarse siempre en un análisis experimental.

3.6. Resumen

En el presente capítulo se ha realizado un análisis de los métodos más destacados de búsqueda por similitud sobre espacios métricos. Primero se ha mostrado una clasificación de los métodos fundamentales, siguiendo diversos criterios. Después se han analizado detalladamente dichos métodos, empezando por los basados en pivotes y siguiendo después con los basados en particiones, acompañándolos de ejemplos clarificadores y de representaciones gráficas descriptivas. Posteriormente se han tratado otros métodos más recientes, algunos con profundidad por usarse posteriormente en esta tesis y otros de manera más breve. Finalmente, se exponen los argumentos que justifican la necesidad de evaluar el rendimiento de los métodos mediante la realización de un análisis experimental.

Capítulo 4

Índice de pivote único

En este capítulo se propone un nuevo método de búsqueda por similitud basado en pivotes, que reduce significativamente el espacio ocupado por el índice, uno de los problemas que habitualmente presentan los métodos basados en pivotes, y que a la vez mantiene un número aceptable de evaluaciones de la función de distancia, hasta el punto de superar en ambos indicadores a otros métodos basados en particiones, a pesar de que estos métodos se caracterizan por generar índices de un tamaño muy reducido. El hecho de que el método propuesto origine índices que ocupan muy poco espacio, permite que sea aplicable a casos reales que realizan el tratamiento de un gran volumen de datos y en los que otros métodos basados en pivotes generan estructuras auxiliares de gran tamaño que los hacen inapropiados en esos escenarios.

En la sección 4.1 se describen los problemas que presentan los métodos basados en pivotes y que han constituido el punto de partida que ha motivado la generación del nuevo método. En la sección 4.2 se mencionan soluciones existentes a esos problemas. En la sección 4.3 se plantean los objetivos que guiarán el desarrollo del nuevo método y que se estudian en las tres secciones siguientes, dando lugar a una serie de soluciones y resultados, fundamentalmente los determinados en la sección 4.6, donde se analizan las características que deben cumplir los pivotes en diferentes colecciones para ser eficaces en el proceso de descarte de cada objeto de la base de datos. En la sección 4.7 se presenta el nuevo método, se comprueba su comportamiento utilizando menor precisión para almacenar las distancias a los objetos y se describe el procedimiento de realización de las búsquedas por rango y de los vecinos más cercanos. En la siguiente sección se presentan y analizan los resultados obtenidos. La última sección hace un resumen del capítulo.

4.1. Debilidades de los métodos basados en pivotes

Como ya se ha indicado en el capítulo introductorio, la búsqueda por similitud sobre espacios métricos puede implementarse de forma trivial comparando el objeto a consultar con la totalidad de los objetos de la base de datos, sin embargo esta alternativa no es viable en la práctica porque la comparación de los objetos supone un elevado coste computacional, debido fundamentalmente al cálculo de las funciones de distancia. El problema se acentúa si la base de datos tiene un gran número de objetos. Por ello se han desarrollado métodos que crean previamente un índice para después utilizarlo en el proceso de búsqueda, con el objetivo de reducir el número de comparaciones entre objetos, aplicando las propiedades de los espacios métricos y la información almacenada en el propio índice. De esta forma se pueden descartar objetos que no cumplen el criterio de búsqueda, sin necesidad de compararlos directamente con la consulta.

El preprocesado del índice puede suponer un coste importante, aunque normalmente no se contabiliza a la hora de evaluar los métodos, centrándose en los factores que se considera que influyen en mayor medida en el rendimiento de las búsquedas, como son principalmente el número de evaluaciones de la función de distancia, ya que sus complejos cálculos incrementan el tiempo de CPU, y también el tamaño del índice, por su incidencia en las operaciones de entrada/salida.

En la mayoría de los escenarios se dan las circunstancias expuestas, es decir, una colección con un gran número de objetos y una función de distancia con un elevado coste computacional. En algún caso se presentan situaciones en las que no puede construirse el índice en la etapa de preprocesado, por ejemplo, cuando los objetos de la colección se obtienen mediante un flujo continuo de datos. Recientemente se han propuesto soluciones que abordan la búsqueda por similitud en esas situaciones [Skopal y Bustos, 2009]. En los restantes casos se realiza el preprocesado de la colección para construir el índice.

La información contenida en el índice está condicionada por el tipo de método utilizado. Los métodos basados en pivotes almacenan las distancias precalculadas de cada objeto en la base de datos al conjunto de pivotes, y estas distancias se utilizan durante la búsqueda para descartar objetos del conjunto resultado. Los métodos basados en clustering particionan el espacio y registran en su índice datos sobre las características de los clusters, para tratar de descartar regiones enteras durante la búsqueda. Necesitan una pequeña cantidad de memoria para almacenar el índice y el tiempo extra de CPU necesario para procesarlo es también muy pequeño. Los métodos basados en pivotes superan a los métodos basados en clustering en términos de cálculos de distancias, pero a cambio de necesitar grandes cantidades de espacio en el índice para almacenar las distancias precalculadas.

La reducción en el número de cálculos de la función de distancia es el principal objetivo de estos métodos, por tratarse del principal componente del coste de la búsqueda, lo que hace que suela utilizarse ese valor asociándolo a la complejidad del método. Pero hay otros factores que afectan al rendimiento global del coste de la búsqueda, como son las necesidades de espacio para almacenar el índice, tanto en memoria principal como secundaria, y el tiempo de CPU adicional requerido para la carga y procesamiento del mismo.

Para poder estimar y comparar el rendimiento de diferentes métodos de búsqueda por similitud, se ha propuesto la siguiente expresión de coste [Chávez y otros, 2001b]:

$$\begin{aligned} T \quad = \quad & \#d \times \textit{complejidad de } d() + \\ & \textit{tiempo I/O} + \\ & \textit{tiempo extra CPU} \end{aligned} \tag{4.1}$$

donde $\#d$ representa el número de evaluaciones de la función de distancia necesarias para resolver una consulta, *tiempo I/O* es el tiempo necesario para transferir el índice de memoria secundaria a memoria principal, y *tiempo extra CPU* es la suma del tiempo que se necesita para procesar la estructura del índice una vez que está en memoria principal y del tiempo necesario para el proceso del resto de operaciones.

Podría simplificarse la expresión anterior considerando despreciables tanto el tiempo I/O como el tiempo extra CPU. Esta aproximación sería válida siempre y cuando el tamaño de las colecciones a indexar fuera relativamente pequeño, con lo que el tamaño de los índices sería también reducido, por lo que el tiempo I/O y el tiempo extra CPU podrían considerarse irrelevantes.

La situación puede ser muy diferente si se trabaja sobre colecciones extensas. En este caso el tamaño del índice puede llegar a ser realmente voluminoso y los factores que antes se desestimaban (tiempo extra CPU y tiempo I/O), ahora pueden convertirse en componentes determinantes de la expresión de coste. Por este motivo en los dominios de aplicación en los que se manipulan bases de datos de gran tamaño, el método de búsqueda por similitud debe considerar la totalidad de los factores de la expresión de coste y no solo al número de evaluaciones de la función de distancia.

Recordando lo ya expuesto en la sección 2.4.1, los métodos basados en pivotes seleccionan un conjunto $P = \{p_1, \ldots, p_m\}$ de m objetos de la base de datos, denominados pivotes. En la fase de indexación se calculan las distancias $d(p_i, x_j)$ de los objetos de la base de datos a los pivotes y se almacenan en una estructura de datos apropiada. Dada una consulta $R(q, r)$, el objeto de consulta q se compara con los pivotes para obtener las distancias $d(q, p_i)$. Para cada elemento $x_j \in U$, se puede obtener una cota inferior de su distancia al objeto de consulta q, usando la desigualdad triangular. El objeto x_j puede descartarse del conjunto resultado, sin

compararlo con el objeto de consulta, si:

$$r < |d(q,p_i) - d(p_i,x_j)| \leq d(q,x_j) \quad \text{para algún pivote } p_i \in P \tag{4.2}$$

La figura 4.1 nos permite recordar el criterio de descarte.

Se considera que la complejidad de la búsqueda es la suma de la complejidad interna, determinada ahora por la comparación de la consulta con los pivotes, y de la complejidad externa, resultado de la comparación de la consulta con los objetos que no pueden descartarse utilizando los datos almacenados en el índice.

El proceso de realización de una búsqueda $R(q,r)$ supone los siguientes pasos:

- Calcular las distancias del objeto de búsqueda q a cada pivote p_i con $i = \{1,\ldots,m\}$.

  Supone realizar m evaluaciones de la función de distancia y es en lo que consiste la complejidad interna.

- Realizar el proceso de descarte utilizando la desigualdad triangular y las distancias $d(p_i,x_j)$ almacenadas en el índice.

  No implica cálculos de la función de distancia, pero sí tiempo adicional, tanto *tiempo I/O* como *tiempo extra CPU*.

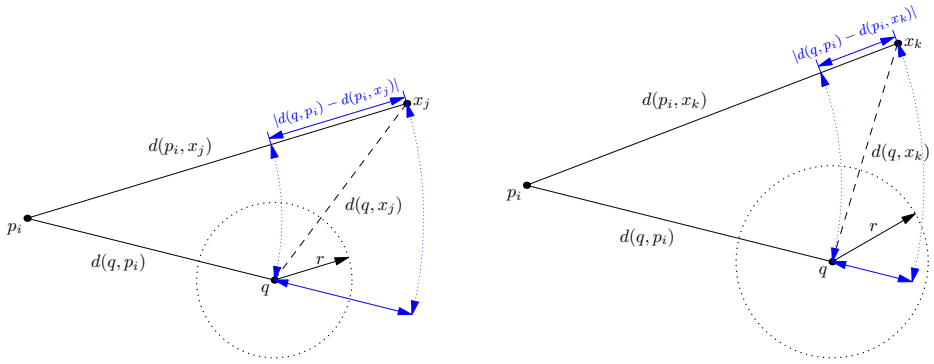
- Comparar los objetos que no han resultado descartados con el objeto de búsqueda q.

  Estas nuevas evaluaciones de la función de distancia determinan la complejidad externa, que está inversamente relacionada con la efectividad lograda por los pivotes en el proceso de descarte, ya que en la medida que los pivotes hayan descartado más objetos, habrá que realizar ahora menos comparaciones de la función de distancia.

4.1.1. Los requerimientos de espacio

En el caso de los métodos basados en pivotes, las consideraciones acerca del espacio son importantes, debido a que el principal inconveniente que buena parte de ellos presentan, es que necesitan crear un índice de considerable tamaño para almacenar las distancias entre los pivotes y el resto de objetos de la base de datos.

Por otra parte, el tratamiento de la información durante las últimas décadas ha abarcado todo tipo de actividades humanas, originando un volumen de datos en constante crecimiento, por lo que las colecciones sobre las que se realizan las búsquedas por similitud son cada vez mayores.



(a) Puede descartarse x_j (b) No puede descartarse x_k

Figura 4.1: Representación del criterio de descarte de objetos en los métodos basados en pivotes.

Las dos circunstancias mencionadas son lo suficientemente relevantes como para resaltar la importancia del tamaño del índice en los métodos basados en pivotes. Podría pensarse que el espacio que ocupa un índice no es un factor determinante hoy en día, dado el bajo coste económico que supone el almacenamiento en memoria secundaria, pero el problema estriba en que en una búsqueda se necesita procesar el índice y eso conlleva costes adicionales, como el coste de tiempo de entrada/salida para realizar el trasvase entre almacenamiento secundario y memoria principal, y el coste de tiempo de CPU para procesar el índice en memoria principal.

Las propuestas de los métodos basados en pivotes, contrastan a menudo su rendimiento en función únicamente del número de evaluaciones de la función de distancia. Incluso destacados métodos como el caso de AESA, reconocido durante más de veinte años como uno de los más eficientes en términos de evaluaciones de la función de distancia [Figueroa y otros, 2009], puede originar tamaños del índice que lo convierten en inaplicable ante colecciones de gran volumen. Incluso a veces, al considerar el coste total de un método de pivotes aplicado a colecciones de gran volumen de datos, se originan unos tiempos de proceso inasumibles, cercanos a los que se obtendrían ante un recorrido secuencial. No es de extrañar que circunstancias de este tipo lleven en ocasiones a cuestionar la viabilidad real de estos métodos [Skopal, 2010].

Pero por otra parte, los métodos basados en pivotes constituyen un campo de investigación atractivo, en el que siguen surgiendo propuestas interesantes [Socorro y otros, 2011] y que presentan la gran fortaleza de que sus algoritmos originan un número reducido de evaluaciones de la función de distancia. Todo ello convierte

en un problema abierto la búsqueda de alternativas para mejorar dichos métodos, siendo una de ellas la reducción del tamaño del índice, ya que se trata de una de las debilidades que presentan.

4.1.2. La selección de los pivotes

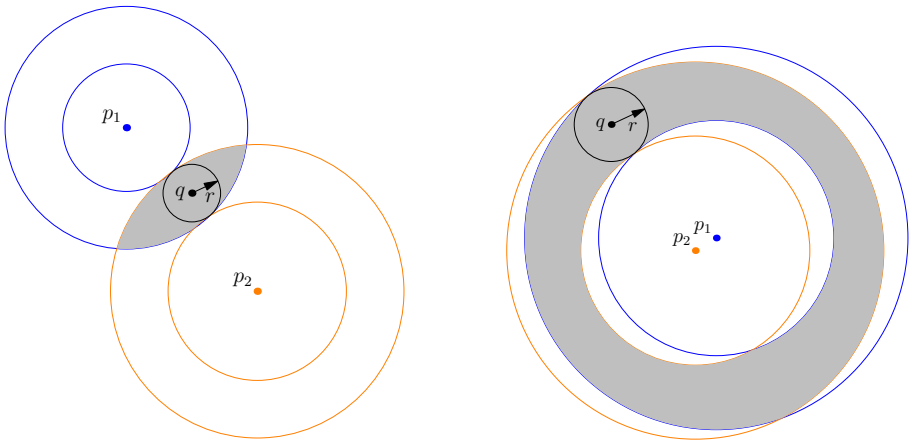
Una cuestión importante relacionada con los métodos basados en pivotes es el criterio seguido para seleccionar los propios pivotes, concretamente qué objetos utilizar y cuál es el número adecuado de ellos. Buena parte de los métodos eligen los pivotes al azar, aunque diversos autores han estudiado el problema, concluido que el conjunto de pivotes seleccionado influye en el rendimiento de los métodos y han propuesto distintas estrategias para la selección de un conjunto eficaz de pivotes [Faragó y otros, 1993; Micó y otros, 1994; Chávez y otros, 2001b; Bustos y otros, 2003; Brisaboa y otros, 2006; Bustos y otros, 2008]. Esto se debe a que la distribución de cada pivote con respecto a los demás, y también con relación a los restantes objetos de la base de datos, tiene una importante incidencia en la capacidad del índice para descartar objetos en el proceso de búsqueda. Esto significa que se descartarán más o menos objetos, lo que conlleva que habrá después que realizar más o menos comparaciones de la función de distancia, por lo que influye en la complejidad externa.

La idea del efecto que ocasiona la distribución de cada pivote con respecto a los restantes, puede describirse con un ejemplo en el que se tiene un conjunto de pivotes que están muy cerca unos de otros. Dada una consulta $R(q, r)$, cada uno de los pivotes determina una zona de exclusión en función de q y de r, como se ha visto en la figura 2.12, de forma que los objetos que pertenecen a ella no pueden descartarse con ese pivote, con lo que, o se descartan mediante otro pivote, o habrá que calcular sus distancias a q, lo que incrementa la complejidad externa. Pero al tratarse de pivotes cercanos, sus zonas de exclusión son también cercanas, hasta el punto de coincidir en buena parte de las mismas, con lo que el efecto de ese conjunto de pivotes queda contrarrestado, haciendo que unos anulen el efecto de los otros. Se puede decir que los pivotes cercanos aportan una información parecida de cara al descarte de objetos, de manera que cuanto más cercanos son dos pivotes, más parecida es su capacidad de descarte, al serlo las zonas de exclusión que originan.

La figura 4.2 muestra un ejemplo de una búsqueda por rango utilizando dos pivotes, realizada sobre el espacio $\mathbb{R}^2$ y usando la distancia L_2.[1] En el lado izquierdo los pivotes están separados, dejando en medio al objeto de búsqueda. La zona de exclusión resultante es la intersección de la correspondiente a cada pivote y aparece

[1] Utilizar el espacio vectorial $\mathbb{R}^2$ con la distancia euclídea, permite representar las ideas de una manera cercana y fácilmente comprensible, pero hay que resaltar la necesidad de diferenciar las características de un espacio vectorial, de las genéricas de un espacio métrico, ya que en el primero se dan algunas propiedades geométricas que no se cumplen en el segundo.

sombreada, estando formada en este caso por una pequeña parte de cada una de ellas, y haciendo que la zona en la que hay que calcular la función de distancia sea solo algo mayor que el círculo de búsqueda. En el lado derecho los pivotes están muy próximos, por lo que sus zonas de exclusión coinciden en buena parte de las mismas, haciendo que la zona de exclusión conjunta sea muchísimo mayor que el círculo de búsqueda. Aunque el segundo pivote reduce esa zona de exclusión, lo hace muy levemente, ya que su efecto queda en buena parte neutralizado por el del primer pivote, y viceversa.



(a) Con pivotes lejanos (b) Con pivotes cercanos

Figura 4.2: Zona de exclusión conjunta usando dos pivotes.

Además de determinar un criterio eficiente para la selección de los pivotes, hay otra cuestión de interés que afecta también al rendimiento de las consultas: el número de pivotes seleccionados. No solo es importante la distribución de los pivotes, sino que finalmente hay que elegir un número determinado de ellos. Cuantos más pivotes se consideren, habrá mayores posibilidades de descartar más objetos, pero hay que calcular también más distancias entre los pivotes y el objeto de búsqueda, lo que implica incrementar la complejidad interna, y posteriormente almacenarlas en el índice, lo que significa que su tamaño será mayor y con él los costes derivados comentados en el apartado anterior. Si el número de pivotes es reducido, se disminuye la complejidad interna y el tamaño del índice, pero también se descartan menos objetos, con lo que habrá que evaluar más veces la función de distancia para comprobar si los objetos no descartados cumplen el criterio de búsqueda, esto es, se incrementa la complejidad externa. En la mayoría de las técnicas de selección de pivotes el número de ellos debe establecerse explícitamente.

Otros métodos como SSS [Brisaboa y otros, 2006] y NR-SSS [Bustos y otros, 2008] determinan el número de pivotes a seleccionar de manera adaptativa, según las características de la colección.

Todo lo comentado refleja la necesidad de considerar criterios que permitan determinar con rigurosidad que los pivotes elegidos sean adecuados y que su número resulte equilibrado.

Las apreciaciones no formales vistas en esta sección, nos alertan ante el impacto que la selección de los pivotes puede tener en el rendimiento de la búsqueda, tanto en la consideración de la manera en la que están distribuidos los pivotes, como en determinar un número de ellos que mantenga el equilibrio entre las complejidades interna y externa. El tamaño de las colecciones influye también en el rendimiento provocando costes adicionales, que en el caso del tratamiento de grandes volúmenes de datos, pueden ser muy significativos, por lo que no deben despreciarse, sino considerarlos como un factor más con el objetivo de que los algoritmos resultantes sean aplicables a casos reales, especialmente al utilizar métodos basados en pivotes, en los que el espacio que necesitan es su principal handicap.

4.2. Soluciones existentes a los problemas de los métodos basados en pivotes

Se han desarrollado varias estrategias tanto para reducir el espacio ocupado por el índice como para seleccionar los pivotes de una manera efectiva, que intentan mantener la eficacia de los métodos de cara al proceso de descarte. Las más relevantes se describen en los apartados siguientes.

4.2.1. Reducción del espacio ocupado por el índice

Las técnicas aplicables para reducir el espacio ocupado por el índice en los métodos basados en pivotes, tienen como objetivo final reducir los factores del tiempo de entrada/salida y del tiempo de CPU.

Las propuestas actuales son [Chávez y otros, 2001b]:

- **Degradar el rango**: Consiste en almacenar las distancias de los objetos a los pivotes con una precisión menor, con lo que se reduce el espacio necesario para el índice. Trabajando con distancias continuas el rango de posibles valores se divide en varios intervalos y el índice almacena el intervalo en el que se

encuentra cada distancia. Las distancias son ahora menos precisas, con lo que la cantidad de objetos desechados será menor.

- **Reducir el nivel de granularidad del nodo**: Está orientada a índices con una estructura arbórea y consiste en interrumpir la creación de subárboles, cuando se ha alcanzado un número suficiente de elementos y los que quedan son un número reducido, que se almacenan todos en el mismo nodo. Realmente lo que supone es que no se almacenan las distancias a todos los pivotes para todos los objetos. Al quedar determinadas áreas sin formar parte del índice, el número de objetos a descartar podría reducirse, pero el índice almacena menos distancias, con la consiguiente reducción de espacio.

- **Reducir el ámbito de acción de los pivotes**: Frente a métodos que almacenan las distancias de todos los objetos de la base de datos a cada uno de los pivotes, esta alternativa consiste en guardar las distancias de cada pivote a un subconjunto de objetos de la base de datos. De esta forma se reduce el ámbito de actuación del pivote. El índice almacena así menos distancias, pero a la vez se reducen las posibilidades de descartar un objeto del conjunto resultado.

Las diferentes técnicas se basan en reducir algún elemento relacionado con el tamaño del índice, pero al hacerlo provocan que la eficiencia del método también decaiga, al originar una pérdida de la capacidad de descarte, lo que provoca un incremento del número de evaluaciones de la función de distancia. De todas maneras, este efecto podría contrarrestarse si se aprovecha el espacio liberado, para incluir nuevos pivotes y ampliar así las posibilidades de descartar más objetos.

4.2.2. Selección efectiva de los pivotes

Buena parte de los métodos realizan una elección aleatoria de los pivotes aunque, como ya se ha comentado, el conjunto de pivotes elegido influye en el rendimiento de las búsquedas.

Diversos trabajos realizan propuestas diferentes para determinar la eficacia de un conjunto de pivotes. En general los autores coinciden en mencionar las características que deberían cumplir un conjunto de objetos para ser considerados buenos pivotes, que son estar lo suficientemente lejanos unos de otros y también del resto de objetos de la base de datos [Yianilos, 1993; Micó y otros, 1994; Brin, 1995].

La condición de lejanía entre los pivotes es intuitiva, pues los pivotes muy cercanos tienen una capacidad muy similar de cara al descarte de objetos, como se ha reflejado en la figura 4.2.

En el caso de espacios de grandes dimensiones, suele ocurrir que las distancias entre los objetos son muy grandes, por lo que la elección aleatoria de pivotes suele conllevar una cierta lejanía entre ellos.

Se ha propuesto también un criterio formal para comparar dos conjuntos de pivotes, basado en la distribución de distancias del espacio métrico [Bustos y otros, 2003]. En el mismo trabajo se muestra que los conjuntos de buenos pivotes siguen la pauta de ser distantes, tanto de los otros pivotes como de los demás objetos.

Estos trabajos están orientados a determinar la eficacia de un conjunto de pivotes para la totalidad de los objetos de una base de datos, pero no estudian cuál es el mejor pivote para un objeto determinado.

En lo referente a las técnicas para seleccionar pivotes, se han realizado varias propuestas. A continuación se mencionan algunas de ellas.

- **MaxSum** [Micó y otros, 1994]: Selecciona de manera incremental un conjunto de objetos que maximizan la suma de las distancias entre ellos.
- **MaxMin** [Vleugels y Veltkamp, 2002]: Consiste en elegir un conjunto de objetos que maximizan la distancia mínima entre ellos.
- **Incremental** [Bustos y otros, 2003]: Elige de forma incremental un conjunto de pivotes que maximizan un criterio propio para comparar dos conjuntos de pivotes.
- **SFLOO** [Hennig y Latecki, 2003]: Selecciona los pivotes en función de minimizar una medida de la pérdida de precisión en la búsqueda.
- **Spacing** [van Leuken y otros, 2006]: Escoge un conjunto de pivotes con una correlación mínima y que maximiza la distancia entre los objetos.
- **Maximun Pruning** [Venkateswaran y otros, 2008]: Selecciona pivotes que maximizan el proceso de descarte en una muestra de consultas.
- **Sparse Spatial Selection (SSS)** [Brisaboa y otros, 2006; Bustos y otros, 2008]: Selecciona un conjunto de pivotes de forma dinámica y que está bien distribuido en el espacio, en el sentido de estar lo suficientemente lejos unos de otros.

La última técnica mencionada se utilizará más adelante en la evaluación experimental previa y en la selección del conjunto de pivotes que se utilizará en la propuesta planteada en este capítulo, por lo que se describe con más amplitud a continuación.

Sparse Spatial Selection (SSS)

Como ya se mencionó en 3.4, *Sparse Spatial Selection (SSS)* es un método de selección de pivotes que garantiza una buena distribución de los pivotes en el espacio, haciendo que los pivotes estén lejos unos de otros, que cada objeto de la base de datos tenga algún pivote relativamente cerca y que, a la vez, el número de pivotes no sea elevado.

La característica de que los pivotes estén lo suficientemente distantes unos de otros, se obtiene considerando que cuando un nuevo objeto se introduce en la base de datos, se elige como pivote si está lo suficientemente distante de los pivotes ya seleccionados. Para ello usa un parámetro α, de manera que un nuevo objeto se convierte en pivote si su distancia a los actuales pivotes es mayor que $M\alpha$, siendo M la distancia máxima entre dos objetos cualesquiera del espacio, y α una constante relacionada con la densidad de los pivotes en el espacio, que toma valores en el intervalo $(0, 1]$, y que normalmente logra que se optimice el número de pivotes si está entre $0,35$ y $0,4$.

Al determinar un criterio de selección de pivotes, realmente se tiene un nuevo método de búsqueda por similitud. Los pivotes en SSS se eligen a medida que los objetos se incorporan a la base de datos, tomando como primer pivote el primer objeto. A partir de él, los restantes pivotes se determinan en función de si su distancia a todos los objetos que forman el conjunto de pivotes es mayor que $M\alpha$, almacenando en el índice las distancias de cada pivote a los objetos de la base de datos, que es lo que se utiliza en el proceso de búsqueda para intentar descartar los objetos. La generación de pivotes finaliza cuando se recubre la totalidad del espacio ocupado por la colección. A partir de este momento ya no será necesario generar más pivotes, aunque la colección crezca de manera considerable.

SSS aporta dos importantes cualidades, por una parte el dinamismo, con lo que se pueden realizar operaciones de manipulación de los objetos, permitiendo construir el índice a medida que se introducen nuevos objetos en la base de datos, ya que incluso puede estar vacía inicialmente, determinando si un objeto será o no un pivote en el momento en el que se introduce en la base de datos, y por otra que es adaptativo, ya que se adecúa a la complejidad de la colección y determina el número de pivotes en función de ella.

Con este método se garantiza que el número de pivotes depende de la complejidad del espacio y no del número de objetos de la colección.

El número de pivotes que se originan depende de las características del espacio y de la distribución de sus objetos, incluso de su orden de aparición, pero la incidencia de este hecho en el rendimiento de las búsquedas es casi inexistente. Los pivotes están distribuidos por el espacio con una cierta uniformidad, estando relativamente lejos

unos de otros, ya que se encuentran al menos a distancia $M\alpha$, pero a la vez ocurre que cada pivote tendrá objetos de la base de datos cercanos a él, a una distancia menor que $M\alpha$.

4.2.3. Número de pivotes a elegir

El número de pivotes a utilizar no ha sido precisamente la cuestión más estudiada en los métodos basados en pivotes. El hecho de que el objetivo principal se centrase en reducir el número de evaluaciones de la función de distancia, ha desplazado a las consideraciones relacionadas con el espacio y con ello, al número de pivotes a incluir.

Evidentemente cuantos más pivotes se utilicen, menor número de comparaciones de la función de distancia se necesitan, aunque siguiendo esa premisa, el espacio necesario para almacenar las distancias entre los pivotes y los objetos puede ser muy elevado.

Un ejemplo es AESA que, al considerar que cualquier objeto puede ser un pivote, cuando se aplica a colecciones muy extensas origina índices de gran tamaño que lo convierten en inadecuado. Aún así AESA es un método importante y que puede ser aplicado a casos reales, obteniendo excelentes resultados si, por ejemplo, las colecciones no son extensas y si los cálculos de las funciones de distancia son realmente costosos en proceso de CPU, como es el caso del tratamiento de las secuencias de ADN.

Como ya se ha mencionado, las condiciones solicitadas a los pivotes, de lejanía entre ellos y además entre los restantes objetos, intentan obtener pivotes que resulten eficientes para la totalidad del conjunto de objetos de la base de datos, en términos de evaluaciones de la función de distancia.

Pero el objetivo de un conjunto de pivotes, en el caso de una búsqueda por rango $R(q, r)$, es determinar si cada objeto de la base datos x_j puede descartarse sin evaluar la función de distancia. Si un objeto no se descarta con un pivote, se prueba con otro. Si definitivamente no se descarta con ningún pivote, lo ideal sería que fuese porque cumple el criterio de búsqueda. La reiterada comprobación del criterio de descarte sobre los objetos, utilizando pivotes que no logran descartarlos, no resulta eficiente. Aunque estas comprobaciones no generan nuevas evaluaciones de la función de distancia, sí consumen tiempo de CPU, que se malgastaría si finalmente no se obtienen resultados positivos.

La mayoría de las técnicas de selección de pivotes requieren que se indique previamente el número de pivotes a elegir, mientras que en el caso de SSS y en NR-SSS el número de pivotes se determina de manera adaptativa según las características de la colección.

K Vantage Points (KVP)

Un trabajo determinado ha explorado el problema del número de pivotes a elegir [Celik, 2002], realizando unas comprobaciones empíricas e incluyendo una propuesta que se mencionan a continuación:

- Comprueba que la eficiencia de los pivotes depende de su distancia al objeto de búsqueda q , resultando que los pivotes descartan más objetos si están cerca o lejos del objeto de búsqueda.
- Aunque se use un número elevado de pivotes, solo una pequeña parte de ellos son efectivos de cara al descarte de cada objeto concreto. Determina, siempre empíricamente, que eligiendo primero los pivotes más cercanos al objeto de búsqueda, se eliminan los objetos de manera más rápida.
- Extiende la idea a la distancia entre cada objeto de la base de datos y los pivotes, para proponer una estructura que almacena para cada objeto, solo los k pivotes más prometedores, en la práctica solo dos pivotes, el más cercano y el más lejano al objeto (*K Vantage Points, KVP*).

4.3. Planteamiento de la propuesta

Los métodos basados en pivotes requieren, para su aplicación efectiva a colecciones reales y voluminosas, que son cada vez más frecuentes en el tratamiento actual de la información, que en su análisis se considere el coste total que originan, especialmente el derivado del espacio ocupado por el índice, que se traduce en tiempo de entrada/salida y tiempo de CPU necesario para procesar el propio índice, y no solo el número de evaluaciones de la función de distancia.

Las condiciones del dominio de aplicación determinan también la importancia de los factores asociados al coste, ya que en unos casos pueden ser más importantes unos que otros, aunque el número de evaluaciones de la función de distancia es siempre determinante. Eso hace que unos métodos puedan ser aplicables en algunas situaciones, pero puede que no lo sean en otros escenarios.

Con el afán de ampliar las posibilidades de estos métodos de cara a su aplicación a diversos entornos de diferentes características, se considera la posibilidad de desarrollar una alternativa que equilibre el coste total, considerando especialmente que las necesidades de espacio del índice sean lineales, pero manteniendo a la vez un rendimiento efectivo en el proceso de búsqueda.

Aunque se han comentado alternativas existentes para la reducción del espacio ocupado por el índice, todas ellas son complementarias, ya que por sí mismas no

suponen una mejora contundente o un cambio fundamental en un método. Por tanto, el objetivo se enfocará en un criterio de selección del conjunto de pivotes que realmente suponga una mejora significativa.

Se han comentado las condiciones que, de forma genérica, se piden a un conjunto de pivotes, en el sentido de lejanía entre ellos y también entre los otros objetos de la base de datos. También se ha hecho hincapié en que, normalmente, se considera que las características de capacidad de descarte de un conjunto de pivotes se refieren a la totalidad de elementos del espacio, y no a cada objeto por separado.

La orientación de la presente propuesta se centra en determinar un conjunto de pivotes que resulten eficaces para descartar cada uno de los objetos de la base de datos, pero también que en conjunto sean válidos para descartar esos objetos, que originen un índice que necesite muy poco espacio para poder aplicarlo a casos reales aunque traten extensos volúmenes de datos, y finalmente, que mantengan un número de evaluaciones de la función de distancia adecuado.

Para que un conjunto de pivotes resulten eficientes para cada uno de los objetos, debe primero favorecer el descarte de cada objeto de manera individual, pero también generar un número reducido de evaluaciones de la función de distancia para el conjunto de objetos, porque podría ocurrir que para el conjunto de objetos originasen demasiadas comparaciones, por ejemplo en el caso límite de producirse un número tan elevado de pivotes que sea cercano al de objetos, con lo que el coste de la búsqueda sería parecido al de una búsqueda secuencial.

4.3.1. Determinación de objetivos

Para desarrollar la propuesta planteada, se marcan los siguientes objetivos:

- Estudiar los factores que contribuyen a incrementar la eficacia de los pivotes para cada objeto de la base de datos.
- Intentar reducir el espacio ocupado por el índice, minimizando el número de distancias que almacena para cada objeto de la base de datos.
- Analizar la eficacia de los pivotes para cada objeto de la base de datos, mediante un análisis experimental con diferentes colecciones de datos.
- Realizar la propuesta de un nuevo método que se base en los resultados anteriores y que logre que el rendimiento de las consultas se mantenga en unos niveles aceptables.

Los objetivos están encadenados, necesitando cada punto los resultados del anterior.

4.4. Estudio de los factores que incrementan la eficacia de los pivotes

Las ideas se plasman para la búsqueda por rango, aunque las conclusiones que se presentan afectan también a la búsqueda de los k-vecinos más cercanos. Se expone un planteamiento basado en el análisis de las diferentes posibilidades de incrementar la eficacia de los pivotes, acompañado de ejemplos.

En una búsqueda $R(q,r)$ la capacidad de descarte de un pivote p_i para un objeto concreto x_j viene dada por la condición del criterio de descarte, expuesta en la página 33 de la sección 2.4.1 y representada ahora en la figura 4.3, que establecía una cota inferior para la distancia de q a x_j:

$$r \ < \ |d(q,p_i) - d(p_i,x_j)| \ \leq \ d(q,x_j)$$

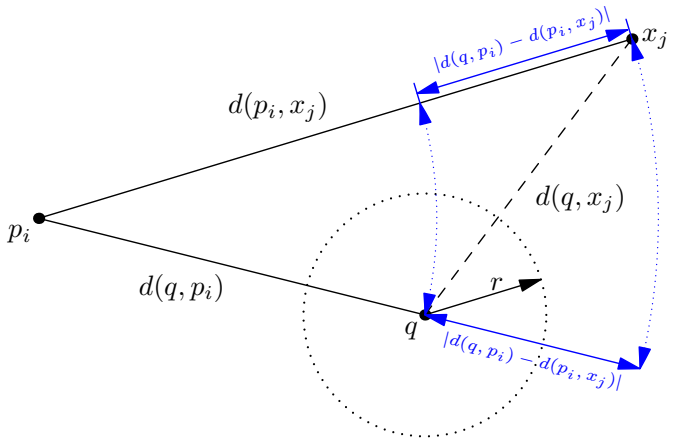


Figura 4.3: Criterio de descarte para $R(q,r)$ usando el pivote p_i.

Si $r \ < \ d(q,x_j)$ entonces el objeto x_j puede descartarse, y la manera de comprobarlo, sin calcular la distancia entre q y x_j, es verificar si la cota inferior es mayor que r, esto es, si $r \ < \ |d(q,p_i) - d(p_i,x_j)|$.

Para incrementar la capacidad de descarte del pivote p_i la cota inferior debe maximizarse, y al hacerlo en función del pivote, se determinará el que resulte más eficiente para descartar x_j. La cota inferior $|d(q,p_i) - d(p_i,x_j)|$ consta del valor absoluto de la diferencia entre dos elementos, por lo que se logra maximizarla si uno

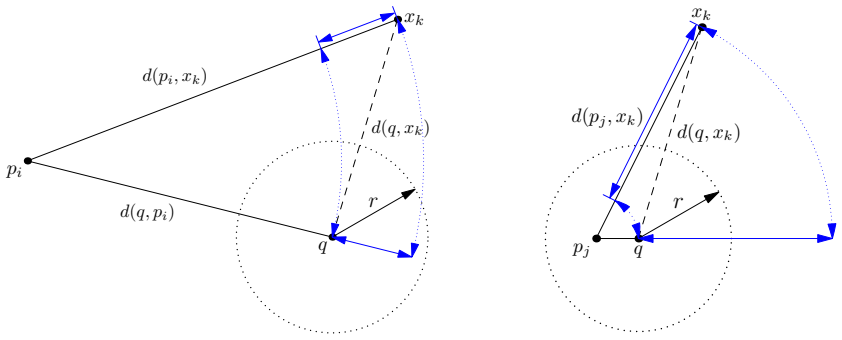
de los elementos tiende a cero mientras que el otro toma el valor más alto posible. En cualquier caso no es suficiente incrementar o reducir un elemento de la cota inferior, ya que si el otro posee un valor similar, ambos se contrarrestan haciendo que el valor de la cota tienda a cero y eso prácticamente anulará las posibilidades de descartar el objeto con el pivote elegido.

A continuación se discuten las diferentes posibilidades que se dan al maximizar la cota inferior.

Minimizar el valor de $d(q, p_i)$

El primer elemento de la cota inferior es $d(q, p_i)$ y minimizarlo implica que el pivote p_i esté muy cerca del objeto de búsqueda q. Además para que el pivote sea efectivo, la distancia $d(q, p_i)$ tiene que ser muy diferente de $d(p_i, x_j)$ para que ambas distancias no se contrarresten. La idea de proximidad del pivote con el objeto de búsqueda es la que se perseguía en el criterio de aproximación utilizado en AESA (página 72). También explica el resultado de la segunda comprobación obtenido en KVP, en el sentido de utilizar primero los pivotes más cercanos a q.

La figura 4.4 muestra esta situación, ya que en el lado izquierdo no puede descartarse el objeto x_k mediante el pivote p_i, pero en el lado derecho, utilizando un pivote más cercano a q como p_j, el mismo objeto x_k resulta descartado.



(a) x_k no puede descartarse con p_i (b) x_k se descarta con p_j

Figura 4.4: Efecto en el criterio de descarte al utilizar dos pivotes sobre q, siendo p_i más lejano a q y p_j más cercano.

Sin embargo, la idea de utilizar un pivote cercano al objeto de búsqueda es de difícil aplicabilidad, más allá de lo ya visto en otros métodos. Por ejemplo,

AESA (página 72) se beneficia de este resultado, pero para ello necesita un buen número de candidatos a pivotes, en este caso todos los objetos de la base de datos, y evidentemente un elevado espacio en el índice. En la exposición de KVP (página 109) se comprueba empíricamente que es mejor elegir primero los pivotes más cercanos a q en la aplicación del criterio de descarte. Pero ocurre que los objetos a buscar surgen en el propio proceso de búsqueda, posteriormente a la creación del índice, y por tanto al momento en el que se determinan los pivotes a utilizar, por lo que centrarse en minimizar la distancia $d(q, p_i)$ no parece vislumbrar más resultados que los ya conocidos.

Maximizar el valor de $d(q, p_i)$

La otra posibilidad de maximizar la cota actuando sobre su primer elemento consiste en maximiza el valor de $d(q, p_i)$, lo que significa que el pivote p_i estará muy lejos del objeto de búsqueda q.

De la misma forma que en el caso anterior, la distancia $d(q, p_i)$ tiene que ser diferente de $d(p_i, x_j)$ para que no se contrarresten. En la figura 4.5 se observa este efecto en el que ambas distancias se contrarrestan. Si se elige un pivote p_i que haga que $d(q, p_i)$ y $d(p_i, x_j)$ sean muy parecidas, la cota inferior tiende a cero y el pivote no permitirá descartar el objeto. Esto ocurre tanto si el pivote está cerca como si está lejos del objeto que se desea descartar. En la figura se muestra, para cada pivote, la zona de exclusión en la que no se puede determinar si el objeto se puede descartar. Los pivotes p_0, p_1 y p_2 no pueden descartar el objeto x_j, ya que para ellos las distancias involucradas en la cota inferior son iguales o muy similares. Sin embargo, x_j puede descartarse eligiendo el pivote p_3 porque las distancias que intervienen en la cota inferior son lo suficientemente diferentes.

Si se maximiza el valor de $d(q, p_i)$ la cota inferior tiende a crecer y esto hace que el pivote tenga más capacidad para descartar un objeto, teniendo en cuenta lo mencionado en el párrafo anterior.

De este caso y del anterior se deduce que los pivotes que pueden resultar efectivos están o muy cerca o muy lejos del objeto de búsqueda, aunque no todos los pivotes de esas características van a ser realmente efectivos.

Minimizar el valor de $d(p_i, x_j)$

El segundo elemento de la cota inferior es $d(p_i, x_j)$. La cota se maximiza si ese valor tiende a cero, lo que conlleva que el pivote estará muy cerca del objeto que se intenta descartar. Evidentemente esto solo será efectivo si hay una diferencia significativa entre $d(p_i, x_j)$ y $d(q, p_i)$, para que ambas no se neutralicen.

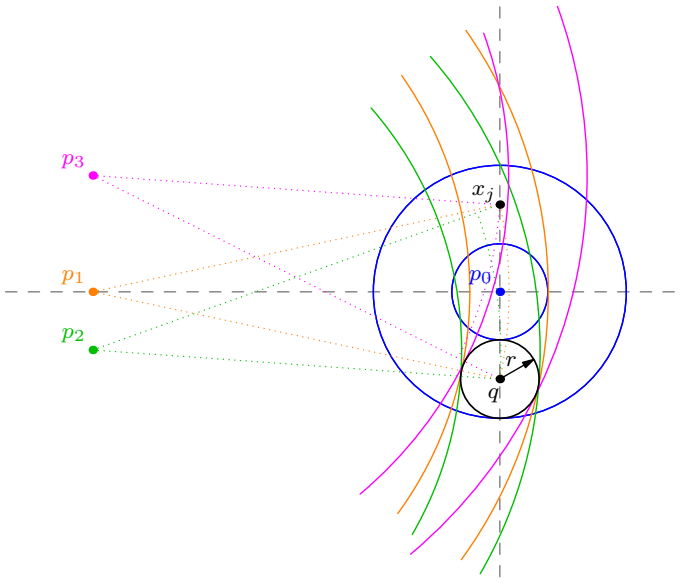


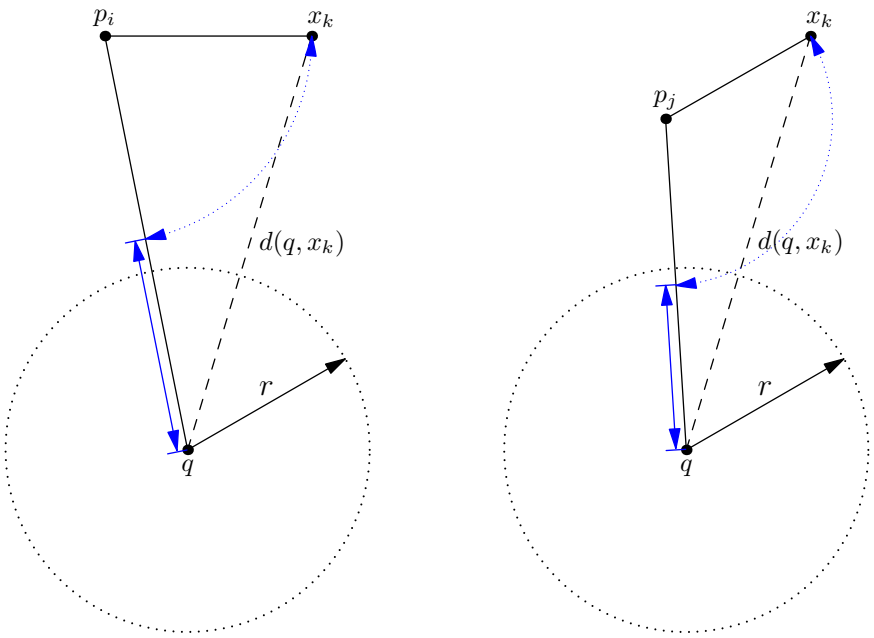
Figura 4.5: Posibilidades de descarte según la elección de pivotes utilizando la cota inferior.

La figura 4.6 muestra un ejemplo en el que el pivote está muy cerca del objeto que se intenta descartar. En el lado izquierdo se tiene un pivote p_i cercano al objeto que se intenta descartar, x_k, y con él se logra. En el lado derecho otro pivote p_j, también cercano a x_k, no consigue descartarlo debido a que el pivote está también demasiado cerca de q y las distancias de la cota inferior se neutralizan. Esto recalca la idea de que para que el pivote sea efectivo, además de estar cerca del objeto que intenta descartar, tiene que estar lo suficientemente lejos del objeto de búsqueda.

Maximizar el valor de $d(p_i, x_j)$

La otra posibilidad de maximizar la cota inferior actuando sobre su segundo elemento, $d(p_i, x_j)$, es hacerlo tan grande como sea posible, mientras que a su vez el otro elemento, $d(q, p_i)$, mantenga un valor muy diferente.

Lo que se obtiene es que si un pivote p_i está muy lejos del objeto que se intenta descartar, x_j, el pivote tendrá grandes posibilidades de descartarlo si las distancias $d(p_i, x_j)$ y $d(q, p_i)$ son lo suficientemente diferentes.



(a) Puede descartarse x_k con p_i

(b) No puede descartarse x_k con p_j

Figura 4.6: Efecto en el criterio de descarte al utilizar un pivote más cercano a x_k .

Conclusiones obtenidas al maximizar la cota inferior

El análisis de las posibilidades de maximizar la cota inferior, formaliza las apreciaciones expuestas en diversos trabajos, en el sentido de que elegir pivotes lejanos a los objetos que se pretende descartar, da buenos resultados, o afirmaciones como que los buenos pivotes son lejanos y a la vez que no todos los pivotes lejanos son efectivos [Bustos y otros, 2003].

Como resumen del estudio de la cota inferior se determina que, tanto los pivotes muy cercanos como los muy lejanos a un objeto de la colección, resultan efectivos para descartar ese objeto, si además se cumple que las dos distancias involucradas en la cota inferior son significativamente diferentes. Por tanto la condición de cercanía o de lejanía es necesaria, pero no es suficiente. Lo mismo se puede concluir acerca de los pivotes cercanos y lejanos a los objetos de búsqueda.

4.5. Reducción del espacio ocupado por el índice

Hasta ahora se ha planteado el estudio desde la óptica de buscar las condiciones para que un pivote sea efectivo para un objeto determinado, resultando ser, en principio, los más cercanos o los más lejanos. Como del objeto de búsqueda no se tienen datos hasta el momento de realizar las consultas, no se hará depender de él la elección de los pivotes.

Para poder disponer de pivotes cercanos y lejanos, se requiere un criterio de selección que obtenga pivotes en esas condiciones, que garantice en definitiva que para cada objeto se tendrán pivotes cercanos y lejanos. Un método como SSS obtiene precisamente pivotes bien distribuidos en el espacio, de forma que se garantiza que cada objeto tendrá pivotes cercanos y lejanos.

Además de que los pivotes sean efectivos para cada objeto, debe tenerse también en cuenta que los pivotes sean efectivos en conjunto, por ejemplo, que no se elijan dos pivotes próximos para intentar descartar el mismo objeto, porque, como ya se ha comentado, aportan casi la misma información, contrarrestando cada uno de ellos el efecto del otro. Esto incide en el interrogante sobre el número de pivotes que son necesarios para cada objeto. Hay dos condicionantes relacionados con esta cuestión: el primero es el hecho de que los pivotes que se encuentran a una distancia intermedia no resultan tan eficientes, y el segundo es que se intenta que el índice ocupe un espacio reducido, preferentemente lineal.

Los dos condicionantes mencionados en el apartado anterior llevan a plantear la hipótesis fundamental de este capítulo, que es que un número muy reducido de pivotes, cercanos y lejanos al objeto a descartar, pueden ser suficientes para descartar con eficiencia los objetos de la base de datos.

4.6. Análisis experimental de la eficiencia de los pivotes

Después de las consideraciones anteriores, se requiere determinar la viabilidad de la hipótesis planteada, para lo que se realizará un análisis que permita contrastar la eficacia para descartar un objeto de la base de datos usando los pivotes cercanos y lejanos a ese objeto. Los factores a evaluar son los más determinantes en la eficiencia de las búsquedas por similitud, el número de evaluaciones de la función de distancia y el tamaño del índice.

Entre las varias cuestiones a concretar de cara a cómo afrontar el análisis, la primera es la elección de un conjunto de pivotes que resulte prometedor de cara al

proceso de descarte. Dado que se necesita disponer de pivotes cercanos y lejanos a los objetos de la base de datos, se requiere un método de selección de pivotes que garantice que aportará pivotes con esas características. Los métodos que obtienen pivotes de manera aleatoria son un ejemplo extremo de técnicas no adecuadas con estos condicionantes. Como ya se ha adelantado en la sección anterior, una alternativa válida es SSS, ya que obtiene pivotes bien distribuidos en el espacio, con lo que para todo objeto de la base de datos habrá siempre un pivote relativamente cerca y otro relativamente lejos del mismo. Por este motivo en este trabajo se usa SSS como criterio de elección de pivotes, aunque se contrastará también el caso de realizar una selección aleatoria.

Otra cuestión importante es establecer un número de pivotes que sea efectivo, lo que entronca con el espacio necesario para el índice, ya que cuanto mayor sea el número de pivotes, mayor será el espacio ocupado. Esto conlleva la necesidad de utilizar un número muy reducido de pivotes. Del estudio de la cota inferior resulta que los pivotes más prometedores son el más cercano y el más lejano, aunque también se ha visto que eso no garantiza siempre buenos resultados. En el análisis se consideran las distancias a esos dos pivotes, pero, dado que se trata precisamente de determinar qué y cuántos pivotes son adecuados para cada objeto de la base de datos, también se probará el efecto con las distancias de los dos más cercanos y de los dos más lejanos a cada objeto.

En definitiva, el análisis realizará primero unas pruebas de contraste, con una elección aleatoria de pivotes y aplicando después SSS en su versión óptima, utilizando en los dos casos las distancias de todos los objetos a cada uno de los pivotes resultantes. Después se elegirán los pivotes mediante SSS para contrastar el efecto de utilizar las distancias a cuatro pivotes, los dos más cercanos y los dos más lejanos, y también a solo dos pivotes, el más cercano y el más lejano.

Una cuestión importante relacionada con el método SSS es que su aplicación depende de un parámetro α que influye en el número de pivotes generados, y que los autores han determinado que da valores óptimos si está entre $0,35$ y $0,4$. Si su valor es mayor, habrá menos pivotes y en consecuencia los pivotes estarán más separados entre sí; si su valor es menor ocurrirá lo contrario, y los pivotes estarán a menor distancia. Esta segunda situación es la que inicialmente interesa en este análisis, ya que se necesitan pivotes cercanos a los objetos, por lo que parece necesario usar un valor de α más bajo que el óptimo.

Se realizaron pruebas preliminares que determinaron que con dos y con cuatro pivotes por objeto, los resultados eran mejores para valores de α entre $0,23$ y $0,3$, ya que con un valor menor se generaban muchos pivotes y los resultados finales empeoraban, y con un valor mayor también empeoraban, pero ahora porque se disponía de pocos pivotes. Por ello se utilizó el valor de $\alpha = 0,25$ para estos casos.

Para realizar el análisis se han considerado las siguientes colecciones incluidas en la librería *Metric Spaces Library*[2]:

- UNIFORM-08 (U08), UNIFORM-10 (U10), UNIFORM-12 (U12), UNIFORM-14 (U14): colecciones de 100.000 vectores de dimensión 8, 10, 12 y 14 respectivamente, distribuidos uniformemente en el cubo unidad.
- ENGLISH: una colección de 69.069 palabras extraídas de un diccionario de inglés.
- NASA: contiene 40.150 imágenes de los archivos de la NASA, representadas por vectores de dimensión 20.
- COLORS: contiene 112.544 histogramas de colores, representados por vectores de dimensión 112.

Las cuatro primeras son sintéticas y están formadas por vectores. La siguiente es real y está formada por palabras. Las dos últimas son reales y están formadas por vectores que provienen de imágenes. El objetivo de esta selección fue comprobar el efecto con colecciones de diferentes características. Las dos últimas tienen realmente naturaleza similar, pero se utilizó COLORS en algún contraste dada la dimensión de sus vectores.

El 90 % de los elementos de las colecciones se ha utilizado como objetos a indexar y el 10 % restante se ha destinado a objetos a consultar, siguiendo el criterio establecido por otros autores [Bustos y otros, 2003]. Se ha utilizando la distancia de edición en la comparación de las colecciones de palabras, mientras que en las de vectores se ha usado la distancia euclídea. Como rango de la búsqueda se usó $r = 2$ para las colecciones de palabras, y en las de vectores, el que obtuviese el $0,01\,\%$ de la base de datos, en promedio para cada consulta.

Los resultados se muestran en la tabla 4.1 que indica el número de pivotes ("#Piv."), el espacio en MB utilizado por el índice ("Esp. (MB)") y el número de evaluaciones de la función de distancia ("#Eval. d") obtenidas al utilizar:

- Una selección aleatoria de pivotes, almacenando todas las distancias de cada objeto a cada uno de los pivotes ("Random").
- SSS, almacenando todas las distancias de cada objeto a cada pivote ("SSS (α óptimo)").
- SSS, almacenando las distancias a los dos pivotes más cercanos y a los dos más lejanos a cada objeto ("SSS ($\alpha = 0,25; 4\ dist$)").

[2] http://sisap.org/Metric_Space_Library.html

- SSS, almacenando las distancias al pivote más cercano y al más lejano a cada objeto ("SSS ($\alpha = 0,25; 2\ dist$)").

Colec.	**Random**			**SSS (α óptimo)**		
	#Piv.	Esp. (MB)	#Eval. d	#Piv.	Esp. (MB)	#Eval. d
U08	85	29,18	211,78	53	18,20	141,43
U10	190	65,23	468,23	176	60,43	367,14
U12	460	157,93	998,13	250	85,83	645,08
U14	1.000	343,33	2.077,44	491	168,57	1.381,64
English	200	27,57	443,85	212	45,06	354,89
Nasa	77	10,61	276,34	55	7,44	168,62

Colec.	**SSS (α = 0,25; 4 dist.)**			**SSS (α = 0,25; 2 dist.)**		
	#Piv.	Esp. (MB)	#Eval. d	#Piv.	Esp. (MB)	#Eval. d
U08	494	2,75	1.231,66	494	1,38	3.094,13
U10	1.461	2,75	3.011,61	1.461	1,38	5.891,64
U12	4.303	2,76	6.803,09	4.303	1,39	9.739,49
U14	10.000	2,78	14.229,20	10.000	1,41	18.160,91
English	3.100	1,91	7.931,30	3.100	0,96	12.373,45
Nasa	871	1,11	1.308,28	871	0,55	1.958,34

Tabla 4.1: Rendimiento de las búsquedas utilizando diferentes estrategias al seleccionar el conjunto inicial de pivotes.

Un estudio de los resultados permite extraer varias conclusiones, la más destacada es que la reducción del número de pivotes utilizados en el proceso de descarte de cada objeto, tiene una notable influencia en el rendimiento de las consultas, concretamente:

- Incrementando el número de evaluaciones de la función de distancia.
- Reduciendo el tamaño del índice.

El resultado no es sorprendente, pero lo interesante es calibrar en qué medida se producen los incrementos y las reducciones, en función del número de distancias a los pivotes almacenadas en el índice para cada objeto.

Se tiene así que en la colección U12 el número de evaluaciones al utilizar dos pivotes es de aproximadamente 15 veces el número de comparaciones que cuando

se utilizan todos los pivotes con SSS ($15 \times 645,08 = 9.676,2 \approx 9.739,49$), pero en cambio el espacio necesario al utilizar todos los pivotes es de aproximadamente 61 veces el ocupado cuando se utilizan solo dos ($61 \times 1,39 = 84,79 \approx 85,83$).

En el caso de U14, una colección más compleja, la diferencia es aún mayor ya que el número de comparaciones cuando se utilizan dos pivotes es de aproximadamente 13 veces el número de evaluaciones necesarias cuando se utiliza la totalidad de los pivotes ($13 \times 1.381,64 = 17.961,32 \approx 18.160,91$), pero el espacio utilizando todos los pivotes es de aproximadamente 119 veces el necesario al utilizar solo dos ($119 \times 1,41 = 167,79 \approx 168,57$).

Se observa también que en las colecciones más complejas, como U14, ENGLISH o NASA, al pasar de 4 a 2 pivotes la eficacia se reduce aproximadamente un 25 %, mientras que el espacio se reduce aproximadamente en un 50 %. Es decir, la pérdida de rendimiento en las evaluaciones de la función de distancia, es mucho menor, concretamente la mitad, que la reducción de espacio necesario para el índice.

Las pruebas dejan constancia de que al reducir drásticamente el número de distancias de los objetos de la colección a los pivotes, por ejemplo a solo dos, el crecimiento del número de evaluaciones de la función de distancia es importante, pero también que, en general, el tamaño del índice decrece en porcentajes mayores a los que se incrementa el número de evaluaciones de la función de distancia. Evidentemente esta estrategia no es aconsejable en escenarios en los que las evaluaciones de la función de distancia son sumamente costosas, pero en casos más generales, donde ese coste es más moderado y en los que se dan unas restricciones en cuanto al tamaño del índice, como disponer de un espacio limitado, su aplicación puede resultar adecuada y obtener consecuencias satisfactorias.

Los resultados abren otra posibilidad, ya que reflejan un comportamiento parecido al de los métodos basados en particiones, en el sentido de necesitar poco espacio a cambio de incrementar el número de evaluaciones de la función de distancia. Se plantea la cuestión de si sería competitivo frente métodos de este tipo que obtuviesen buenos resultados.

Todo lo anterior avala la posibilidad de que reducir drásticamente el número de pivotes a solo dos, es factible si se desean disminuir las necesidades de espacio del índice. Cabe destacar que el método KVP, ya expuesto en la página 109, plantea precisamente un índice en el que se almacena para cada objeto, las distancias a sus pivotes más cercano y más lejano, aunque allí la idea no se basa en ningún formalismo, ni se realiza ningún análisis experimental que avale el motivo de usar esos pivotes, ni que justifique que se usen dos u otro valor.

Para dar mayor rigurosidad al presente análisis hay otra cuestión que debe contrastarse; se trata de si es más eficaz utilizar el pivote más cercano o el más lejano para cada objeto. El método de selección de pivotes resulta crucial para contestar

a esta cuestión, pero determinado ya ese punto, hay que comprobar si realmente es interesante utilizar las distancias a dos pivotes, si hay diferencias importantes en usar uno o dos, y en ese caso, cuál sería el más recomendable. Para ello se contrastó la efectividad de ambos pivotes en el criterio de descarte de los objetos.

La figura 4.7 muestra, para cada colección, los porcentajes de los objetos que no son descartados ("No descartados"), los descartados solo por su pivote más cercano ("Solo por el más cercano"), los descartados solo por su pivote más lejano ("Solo por el más lejano") y los descartados por ambos a la vez ("Por ambos"). En la figura se observa que la utilización del pivote más cercano origina que se descarten la mayoría de los objetos, aunque un gran porcentaje puede ser descartado por la utilización tanto del pivote más cercano como del más lejano. Solo en unos pocos casos el objeto resulta descartado por su pivote más lejano y no por el más cercano.

Con el fin de analizar estos datos y entender en qué casos se descarta el objeto solo por su pivote más lejano, se calculó la media y la desviación estándar de la distribución de distancias entre los objetos de cada colección. Posteriormente, para los casos en el que los objetos solo fueron descartados por su pivote más cercano ("Más cercano"), solo por el más lejano ("Más lejano") y por ambos ("Ambos"), se calculó la distancia media de los objetos en cada grupo, a sus pivotes más cercano y más lejano. Los resultados aparecen reflejados en la tabla 4.2. Las distancias de los objetos a los pivotes se muestran en puntuaciones estándar (z) para poder realizar su comparación. Los pivotes más cercano y más lejano se han obtenido usando SSS con $\alpha = 0,25$, siguiendo el criterio ya mencionado.

Colec.	μ	σ	**Más cercano**		**Más lejano**		**Ambos**	
			$zd_{cer.}$	$zd_{lej.}$	$zd_{cer.}$	$zd_{lej.}$	$zd_{cer.}$	$zd_{lej.}$
U08	1,5086	0,2452	-4,3989	0,9845	-4,1949	1,3923	-4,3989	1,4331
U10	1,4032	0,2456	-3,7182	2,2671	-3,5147	2,6743	-3,7182	2,6743
U12	1,2652	0,2450	-3,0008	3,5298	-2,7151	3,9788	-3,0416	3,9380
U14	1,1244	0,2469	-2,3669	4,6804	-1,9214	5,0855	-2,4480	5,0450
English	8,3176	2,0260	-2,3335	4,6113	-1,9040	5,1591	-2,2742	4,9617
Nasa	1,2342	0,3424	-2,2611	3,1419	-2,0859	2,9083	-2,1735	2,7623

Tabla 4.2: Puntuaciones estándar de las distancias a los pivotes más cercano y más lejano.

Tomando la colección English como ejemplo, se observa que la distancia media entre dos palabras es $\mu = 8,31$, y la desviación típica es $\sigma = 2,02$. Hay que tener en cuenta que, aunque la distribución de distancias es normal, las puntuaciones estándar se calculan sobre distancias de los objetos descartados a sus pivotes correspondientes. Un objeto será descartado por su pivote más cercano, cuando la

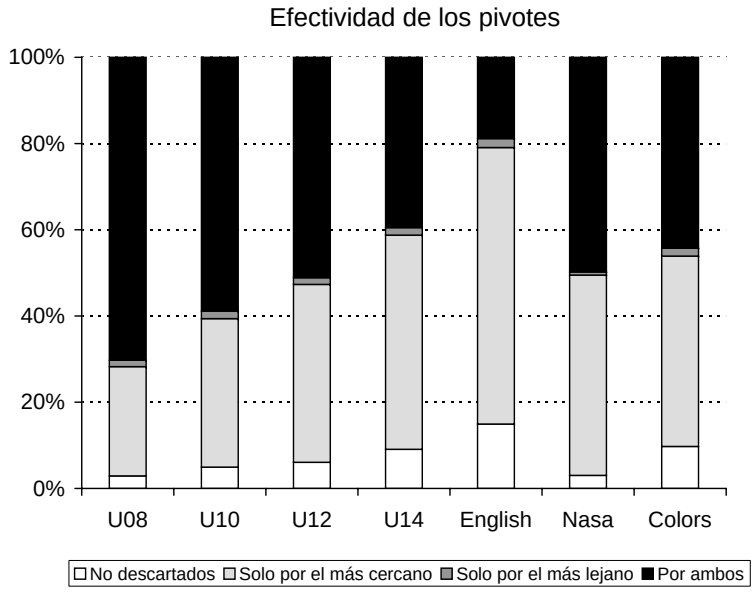


Figura 4.7: Porcentajes de objetos descartados por el pivote más cercano, por el más lejano y por ambos.

puntuación estándar de su distancia a ese pivote es de aproximadamente $-2,33$. Sin embargo, cuando esa puntuación está cercana o es inferior a $-1,9$ y la puntuación estándar de la distancia al pivote más lejano es aproximadamente $5,1$, el objeto va a ser descartado únicamente por el más pivote más lejano. En las dos últimas columnas se muestran las puntuaciones estándar de las distancias a los pivotes más cercano y más lejano, para los objetos que pueden ser descartados por ambos.

Se observa que a medida que la complejidad de las colecciones sintéticas crece, se reducen las posibilidades de descartar el objeto mediante su pivote más lejano. Estos resultados son acordes con los obtenidos en la figura 4.7.

Estas puntuaciones aportan un umbral para cada colección que nos permite decidir, para cada objeto de la base de datos, si será el pivote más cercano o el más lejano, el que tenga más posibilidades de descartarlo.

Como conclusión, el pivote más cercano es el que tiene mayor capacidad para descartar un objeto, y en aquellos casos en los que no está suficientemente cerca, interesa tratar de utilizar el pivote más lejano, si realmente está lo suficientemente lejos. Las puntuaciones estándar de la distancia al pivote más cercano y al pivote más lejano, pueden dar una guía de cuando es adecuado utilizar el pivote más lejano para cada colección. En caso de duda, porque la distancia de ambos podría estar

por encima o por debajo de estos umbrales, la opción más fiable es usar el pivote más cercano.

4.7. Índice de pivote único

De los estudios realizados en las anteriores secciones, se extraen una serie de conclusiones:

- Se ha determinado que el pivote más cercano y el más lejano a cada objeto de la base de datos, son los más prometedores para descartar ese objeto.
- Que en circunstancias en las que la prioridad es reducir el espacio ocupado por el índice, el hecho de considerar un índice formado por un número muy reducido de pivotes para cada objeto de la base de datos, puede ser una alternativa viable. En este caso habría que estudiar si puede resultar competitivo frente a los métodos basados en particiones.
- Se ha analizado en qué condiciones funciona mejor el pivote más cercano y el más lejano para cada colección evaluada. Aunque la aplicación de este resultado requiere un procesado previo de la colección que determine los parámetros que la caracterizan, haciendo posible que se utilicen esos parámetros en futuras búsquedas, es una circunstancia frecuente que las búsquedas por similitud se realicen sobre conjuntos con sus características completamente determinadas.

Los análisis realizados muestran la viabilidad de utilizar para cada objeto de la base datos las distancias a dos pivotes, el más cercano y el más lejano al objeto. El estudio efectuado para determinar la capacidad de descartar objetos que tienen cada uno de esos pivotes, ha clarificado que es muchísimo mayor en el caso del pivote más cercano, hasta el punto de que en casi la totalidad de las situaciones, los objetos que resultan descartados lo son, o solo por su pivote más cercano, o por ambos. Esto plantea la posibilidad de almacenar en el índice únicamente la distancia a un pivote. Pero además en el estudio se han establecido unas condiciones que determinan un criterio de elección del mejor pivote para descartar un objeto, basándose en las puntuaciones estándar.

En función de todo lo anterior, se propone un método basado en pivotes que almacena en el índice, para cada objeto de la base de datos, solo una distancia a un pivote, que será la correspondiente al pivote más cercano o al más lejano, en función de que cumpla las características determinadas previamente para la colección y reflejadas en las puntuaciones estándar.

El criterio de elección de los pivotes es un aspecto fundamental para la eficiencia del método, ya que debe garantizar las siguientes condiciones:

- Generar un conjunto de pivotes bien distribuidos en el espacio, de manera que se obtengan pivotes tanto cercanos como lejanos a los objetos.
- Garantizar que el número de pivotes no es ni demasiado grande, para no incrementar la complejidad interna, ni demasiado pequeño, para que no aumente la complejidad externa.

En la selección de pivotes se ha utilizando el método SSS con el valor $\alpha = 0,25$, porque garantiza las condiciones impuestas, como ya se ha explicado en la sección anterior.

La estructura del índice tiene que almacenar, para cada objeto de la base de datos, el identificador del pivote asociado y la distancia del objeto a dicho pivote. La figura 4.8 representa dicha estructura. Cabe destacar que los pivotes asociados a los diferentes objetos forman un conjunto que tiene que estar limitado en número y a la vez resultar efectivo de cara al proceso de descarte. SSS garantiza estas propiedades.

Ya que el método propuesto utiliza un único pivote para cada objeto de la base de datos, nos referiremos a él como *Índice de Pivote Único (UPI).*

x_1	...	x_n
p_k	...	p_l
$d(x_1, p_k)$	...	$d(x_n, p_l)$

Figura 4.8: Estructura del índice en UPI.

4.7.1. Almacenamiento de las distancias con menor precisión

Dado que uno de los principales objetivos es reducir el espacio del índice, se puede utilizar una precisión menor para almacenar los datos. Concretamente pueden almacenarse los identificadores de los pivotes con $\log_2(m)$ bits, siendo m el número de pivotes, por lo que se necesita menos de un byte en la mayoría de los casos. Para la distancia al pivote podría utilizarse un número variable de bits, perdiendo con ello alguna precisión.

La tabla 4.3 muestra los resultados obtenidos con dos configuraciones diferentes del método. En la primera, ("UPI (4 bytes)"), los identificadores de los pivotes se

almacenaron utilizando el número mínimo de bits y para las distancias se usaron los 4 bytes necesarios para precisión flotante. En el segundo, ("UPI (2 bytes)"), para los identificadores de los pivotes se utilizó también el número mínimo de bits, pero las distancias se almacenaron utilizando 2 bytes, es decir, la mitad de la precisión. Para cada opción se muestra el espacio utilizado por el índice en MB ("Esp. (MB)") y el número de evaluaciones de la función de distancia ("#Eval. d"). Puede observarse en la tabla que almacenar las distancias con menor precisión, apenas provoca una penalización en el número de comparaciones de la función de distancia, mientras que el espacio logra reducirse aproximadamente a la mitad del necesario cuando se almacenan las distancias usando precisión flotante.

Colec.	**UPI (4 bytes)**		**UPI (2 bytes)**	
	Esp. (MB)	#Eval. d	Esp. (MB)	#Eval. d
U08	0,4311	3.094,13	0,2594	3.095,72
U10	0,4348	5.891,64	0,2631	5.893,84
U12	0,4456	9.739,49	0,2740	9.741,91
U14	0,4673	18.160,91	0,2957	18.164,03
English	0,3083	12.373,45	0,1897	12.373,45
Nasa	0,1757	1.958,34	0,1068	1.958,72

Tabla 4.3: Espacio y evaluaciones de la distancia según la precisión utilizada.

El número de bits necesarios para almacenar las distancias, depende del rango de valores que presenta la función de distancia, por lo que el método puede ser parametrizado para utilizar un determinado número de bits, en función de las características de la función de distancia y de la tolerancia a la pérdida de precisión.

4.7.2. Búsqueda por rango

El índice utilizado en UPI solo almacena una distancia para cada objeto de la base de datos, la correspondiente al único pivote asociado a cada objeto. Por lo tanto, en el proceso de realización de una búsqueda por rango solo se puede utilizar una única distancia para intentar descartar cada objeto de la base de datos.

Dada una consulta $R(q, r)$, para un objeto de la base de datos x_j que tiene como pivote asociado p_i, se calcula la distancia del objeto de búsqueda al pivote, $d(q, p_i)$, y con la distancia almacenada en el índice, $d(p_i, x_j)$, se calcula el valor de la cota inferior para comprobar después si el objeto de la base de datos puede descarte, aplicando:

$$r \ < \ |d(q, p_i) - d(p_i, x_j)| \ \leq \ d(q, x_j)$$

Si el objeto x_j no puede descartarse utilizando su pivote asociado p_i, hay que calcular su distancia al objeto de búsqueda, porque no se dispone de la distancia a otro pivote con la que obtener otra cota inferior para ese objeto.

Es inevitable que se incremente el número de evaluaciones de la función de distancia con respecto a otros métodos basados en pivotes. El método basa toda su eficiencia en la elección de una buena cota inferior para cada objeto de la base de datos, usando el pivote más cercano o el más lejano, pero dado que solo se usa un pivote por objeto, solo se dispondrá de una cota inferior para cada objeto. La selección de los pivotes, siempre decisiva en estos métodos, es aquí aún más determinante, de ahí la necesidad de disponer de un método que garantice pivotes cercanos y lejanos a cada objeto, como es SSS.

La figura 4.9 refleja el proceso de búsqueda por rango utilizando UPI. Se tiene una consulta $R(q, r)$ y dos objetos de la base de datos, x_j con pivote p_i, y x_k con pivote p_l. Las distancias entre cada objeto y su pivote asociado están almacenadas en el índice. Las distancias entre el objeto de búsqueda q y los pivotes se calculan en el proceso de búsqueda, incrementando la complejidad interna.

Para el objeto x_j la cota inferior es mayor que el rango de la consulta, con lo que el objeto puede descartarse. Para x_k la cota inferior es menor que r por lo que no puede descartarse. Para comprobar si el objeto cumple el criterio de búsqueda no queda más alternativa que calcular su distancia entre el objeto x_k y el objeto de búsqueda, $d(q, x_k)$. En este caso ocurre que el objeto x_k cumple el criterio de búsqueda.

4.7.3. Búsqueda de los vecinos más cercanos

En los problemas en los que se utiliza la búsqueda por similitud, la búsqueda de los vecinos más cercanos es una operación útil y frecuente. Se ha mencionado que este tipo de búsqueda puede resolverse a partir de la búsqueda por rango, sin embargo, existen algoritmos para resolver este tipo de consultas sin recurrir a la búsqueda por rango. A continuación se describe el procedimiento para la realización de dicha búsqueda utilizando UPI.

La estructura del índice propuesta, mostrada en la figura 4.8, asocia un pivote a cada uno de los objetos de la base de datos, por ejemplo, al objeto x_j le corresponde el pivote p_i, siendo este el pivote más cercano o más lejano a x_j. Dada una búsqueda de los k-vecinos más cercanos a un objeto q, $kNN(q)$, la existencia de un pivote para cada objeto de la base de datos establece una cota inferior para la distancia entre el objeto de búsqueda y el objeto de la base de datos, esto es, se tiene una cota inferior para la distancia entre q y x_j:

$$|d(q, p_i) - d(p_i, x_j)| \leq d(q, x_j)$$

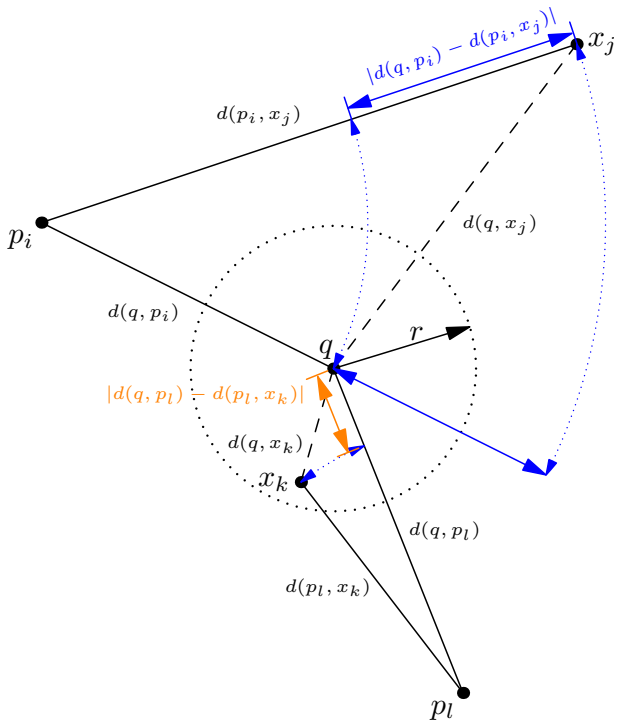


Figura 4.9: Búsqueda por rango en UPI.

Para realizar la búsqueda $kNN(q)$, primero se ordenan los objetos de la base de datos en función del valor de la cota inferior de su distancia al objeto de búsqueda. Seguidamente se calculan las distancias de q a cada objeto de la base de datos, empezando por el objeto al que le corresponde una cota inferior de menor valor y siguiendo en orden ascendente según el valor de la cota.

De esta manera se va creando una lista de objetos candidatos a vecinos más cercanos a q. Una vez que se tienen k candidatos en la lista, se siguen calculando las distancias de los siguientes objetos de la base de datos al objeto de búsqueda, porque un objeto que tenga una cota inferior más alta que cualquiera de los actuales candidatos a vecinos más cercanos, puede estar realmente más cerca de q que alguno de esos candidatos.

El procedimiento finaliza cuando, una vez que se tienen k candidatos, ocurre que al comparar q con el siguiente objeto, resulta que la cota inferior de la distancia

entre ellos es mayor que la distancia a la que se encuentra de q su actual k-ésimo vecino más cercano. En ese momento la lista de candidatos está formada por los k-vecinos más cercanos al objeto de búsqueda y el proceso finaliza.

El proceso de búsqueda kNN descrito se basa en la ordenación de los objetos de la base de datos, lo que puede suponer un problema al utilizar colecciones de gran tamaño. Una alternativa que evita la ordenación completa de los objetos en función de su cota inferior, que presenta complejidad lineal y que además mantiene un número reducido de evaluaciones de la función de distancia, consiste en distribuir los objetos en una serie de intervalos en función de los valores de las cotas inferiores [Ares y otros, 2010]. Con esto se logra una ordenación parcial entre los objetos de diferentes intervalos, ya que todos los objetos pertenecientes a un intervalo presentan una cota inferior menor que la de cualquier otro objeto perteneciente a un intervalo posterior.

4.8. Evaluación experimental

Los indicadores a evaluar para comprobar la eficiencia del método, son pues el tamaño del índice y el número de evaluaciones de la función de distancia, que deben contrastarse frente a los obtenidos en un método basado en particiones, por lo que se utilizará el método Lista de Cluster ya que es uno de los más representativos de la efectividad de estos métodos.

Un estudio completo requiere que el nuevo método se compare con cualquier propuesta basada en pivotes que siga un planteamiento parecido. Dado que KVP es una propuesta anterior [Celik, 2002] conviene establecer las diferencias principales de UPI con relación a KVP:

- KVP aporta una propuesta basada en un planteamiento empírico. En UPI se llega a una propuesta después de un estudio de las características de la cota inferior y de un análisis experimental.
- Ya que se ha demostrado que tanto los pivotes más cercanos como los más lejanos a un objeto de la base de datos, son los que auguran un mejor resultado de cara a descartar ese objeto, un método que aproveche ese resultado debe garantizar que realmente utiliza pivotes cercanos y lejanos.

  En la propuesta de KVP no hay ningún planteamiento de elección de pivotes. En UPI se utiliza un exitoso método de selección de pivotes, SSS, desarrollado en el Laboratorio de Bases de Datos de la UDC y ampliamente contrastado en estudios posteriores, que genera pivotes bien distribuidos en el espacio, de forma que garantiza que siempre se obtiene un pivote cercano y otro lejano a cualquier objeto de la base de datos.

El hecho de que con SSS se pueda determinar la densidad de los pivotes obtenidos, dependiendo del valor del parámetro α, lo hace especialmente adecuado para el presente caso, en el que es necesario que se disponga de pivotes lo suficientemente cerca y lejos de cada objeto. Esto se logró determinando empíricamente el valor más adecuado del parámetro para que los resultados fueran adecuados, como se explicó en la argumentación acerca de la elección del valor de α en la sección 4.6, que llevó a determinarlo en $0,25$ para así generar más pivotes y garantizar una mayor cercanía y lejanía a los objetos.

- El objetivo que se persigue con UPI es reducir en lo posible el espacio utilizado por el índice, lo que ha llevado a considerar el pivote más adecuado para descartar cada objeto de la base de datos, después de estudiar en profundidad las condiciones que debe reunir ese pivote, y a almacenar en el índice únicamente la distancia a ese pivote para cada objeto.

  Esto ha originado un método que necesita espacio sublineal y es capaz de compararse con los métodos basados en particiones, inicialmente mucho mejores que los de pivotes en lo referente al tamaño del índice.

En el análisis experimental se ha considerado únicamente la búsqueda por rango, ya que en este caso, la búsqueda de los vecinos más cercanos sigue las mismas pautas que la búsqueda por rango, sin cambios estructurales ni modificaciones que supongan una orientación diferente.

La experimentación se ha efectuado sobre las colecciones ya mencionadas en la sección anterior, utilizando también el criterio de que el 90 % de los elementos de las colecciones se ha utilizado como objetos a indexar, mientras que el 10 % restante se ha destinado a objetos a consultar. Para los rangos de la búsquedas también se han seguido las mismas pautas que en la sección anterior.

En las pruebas se ha incluido el método SSS con su configuración óptima, porque, aunque realiza un número mucho menor de comparaciones, constituye una buena base de referencia tanto para el número de evaluaciones de la distancia como para la cantidad de espacio utilizado. En definitiva, se han comparado los resultados de UPI con los siguientes métodos:

- SSS con el espacio necesario para su configuración óptima, referenciado como "SSS (α óptimo)".
- KVP almacenando para cada objeto la distancia a sus pivotes más cercano y más lejano, esto es, $k = 2$, indicado como "KVP ($k = 2$)".
- Lista de Clusters. Se ha elegido este método por ser un ejemplo representativo de las soluciones basadas en particiones y porque una de sus características es el poco espacio que requiere.

Colec.	SSS (α óptimo)		KVP (k=2)		UPI		Lista de Clusters	
	Esp.	#Ev. d	Esp.	#Ev. d	Esp.	#Ev. d	Esp.	#Ev. d
U08	18,196	141,43	1,375	8.724,62	0,259	3.095,72	0,364	6.139,21
U10	60,426	367,14	1,379	13.063,63	0,263	5.893,84	0,371	11.264,98
U12	85,832	645,08	1,390	18.955,94	0,274	9.741,91	0,371	17.273,55
U14	168,573	1.381,64	1,412	28.502,26	0,296	18.164,03	0,371	28.253,04
ENGLISH	45,055	354,89	0,960	18.305,43	0,190	8.872,37	0,265	7.885,79
NASA	7,444	168,62	0,555	2.676,93	0,107	1.958,72	0,143	2.027,08

Tabla 4.4: Comparación de UPI con otros métodos en valores reales (espacio en MB).

La tabla 4.4 muestra los resultados obtenidos para las distintas colecciones por los diferentes métodos, reflejando sus valores reales. Para cada método aparece primero el espacio ocupado en MB ("Esp.") y después el número de evaluaciones de la función de distancia ("#Ev. d").

Para dar una idea más precisa de la variación entre los métodos, los mismos resultados de espacio ("Esp. %") y evaluaciones de la función de distancia ("Ev. %"), aparecen ahora en porcentajes en la tabla 4.5, considerando que los valores de UPI representan el 100 %. Así por ejemplo, en la colección U08 para el número de evaluaciones de la función de distancia en Lista de Clusters, se obtiene un 198,31 % lo que indica que prácticamente duplica a las obtenidas con UPI, y eso es lo que ocurre, ya que se pasa del valor de 3.095,72 en UPI al de 6.139,21 en Lista de Clusters.

Colec.	SSS (α óptimo)		KVP (k=2)		UPI		Lista de Clusters	
	Esp. %	Ev. %	Esp. %	Ev. %	Esp. %	Ev. %	Esp. %	Ev. %
U08	7.014,76	4,57	530,15	281,83	100	100	140,44	198,31
U10	22.966,74	6,23	524,10	221,65	100	100	141,01	191,13
U12	31.325,44	6,62	507,19	194,58	100	100	135,40	177,31
U14	57.008,25	7,61	477,34	156,92	100	100	125,46	155,54
ENGLISH	23.750,82	4,00	506,27	206,32	100	100	139,75	88,88
NASA	6.969,85	8,61	519,38	136,67	100	100	133,61	103,49

Tabla 4.5: Comparación de UPI con otros métodos en porcentajes si en UPI se da el 100 %.

Finalmente en la tabla 4.6 se muestra el número de objetos de cada colección y el número de evaluaciones de la función de distancia, tanto en porcentajes sobre el total de cada colección ("Ev. %"), como mostrando su valor real ("#Ev. d"). Por ejemplo, utilizando UPI en U10 se generan 5.893,84 comparaciones, lo que supone un 5,89 % del total de objetos de la colección.

Colec. - Obj.	SSS (α óptimo)		KVP (k=2)		UPI		Lista de Clusters	
	Ev. %	#Ev. d	Ev. %	#Ev. d	Ev. %	#Ev. d	Ev. %	#Ev. d
U08 - 100.000	0,14	141,43	8,72	8.724,62	3,10	3.095,72	6,14	6.139,21
U10 - 100.000	0,37	367,14	13,06	13.063,63	5,89	5.893,84	11,26	11.264,98
U12 - 100.000	0,65	645,08	18,96	18.955,94	9,74	9.741,91	17,27	17.273,55
U14 - 100.000	1,38	1.381,64	28,50	28.502,26	18,16	18.164,03	28,25	28.253,04
Eng. - 69.069	0,51	354,89	26,50	18.305,43	12,85	8.872,37	11,42	7.885,79
Nasa - 40.150	0,42	168,62	6,67	2.676,93	4,88	1.958,72	5,05	2.027,08

Tabla 4.6: Evaluaciones de la función de distancia en porcentajes sobre el total de objetos de las colecciones ("Ev. %") y en valores reales ("#Ev. d").

4.8.1. Discusión

El análisis de los resultados refleja una importante mejora de UPI frente a KVP en todos los indicadores, una mejora significativa frente a LC en prácticamente todos los casos, y como era esperado, un incremento muy sustancial del número de evaluaciones de la función de distancia de UPI frente a SSS, pero a cambio de lograr una enorme reducción del tamaño del espacio del índice.

Los valores obtenidos con UPI en el número de evaluaciones de la función de distancia son de 1.958,72 para una colección de 40.150 objetos como Nasa, de 8.872,37 para English con 69.069 objetos y porcentajes de 3,1 % para U08, de 5,8 para U10 y de 9,7 % para U12, todas colecciones de 100.000 objetos. Este número de evaluaciones indican valores asumibles. Solamente en el caso de U14 el porcentaje sube a un 18,1 %, un valor aún aceptable pero que empieza a acercarse al límite del 20 % que es el que se utiliza como estándar de facto en los problemas relacionados con almacenamiento y optimización (indexación, zonas de desbordamiento, etc.). Los costes de procesar porcentajes menores a un 20 % pueden considerarse aceptables y UPI los obtienen en todas las colecciones analizadas, incluso a veces con porcentajes menores que el 5 % (tabla 4.6).

Con respecto al espacio del índice, uno de los principales objetivos planteados, los resultados de la tabla 4.4 indican unos valores muy bajos, ya que para colecciones de 100.000 objetos se generan índices que no llegan a los 300 KB, y en el caso de Nasa son de aproximadamente 100 KB. La diferencia en este punto con respecto a los restantes métodos analizados es muy destacable, mejorando incluso a Lista de Clusters en todos los casos.

La comparación de UPI con SSS refleja un altísimo rendimiento de SSS en el proceso de descarte con lo que el porcentaje del número de evaluaciones de la función de distancia calculadas en el proceso de búsqueda, es como mucho el 1,38 % del total de la colección, mientras que en UPI ese porcentaje está entre un 3 y un 9,7 %, salvo en U14 y en English donde es de 18,16 y 12,85 % respectivamente (tabla 4.6). Aunque los porcentajes de UPI son aceptables, quedan muy lejos de los obtenidos por SSS en este punto, por lo que si la función de distancia a utilizar es

particularmente costosa, UPI no sería la alternativa más recomendable. Por contra, el espacio de los índices ocupado por UPI es muy reducido, casi se podría decir que despreciable frente al tamaño que ocupan en SSS, dándose valores de 0,2 MB en UPI frente a valores de 60, 85 y 168 MB en SSS. La tabla 4.5 refleja que los índices de SSS pueden incrementarse en porcentajes de decenas de miles con respecto a los de UPI. En el caso de las comparaciones de funciones de distancia el porcentaje de reducción de SSS sobre los valores de UPI está entre el 4 y el 8,6 %. Esto indica que el factor de crecimiento del tamaño del índice en SSS es muchísimo mayor que el correspondiente al número de evaluaciones de la función de distancia en UPI. Por todo ello UPI se convierte en una alternativa a tener en cuenta si la prioridad son las consideraciones de espacio, ya que rompe la tendencia general de los métodos basados en pivotes en relación al elevado tamaño de sus índices.

UPI supera de manera importante a KVP en todos los indicadores y en todas las colecciones analizadas. Las diferencias en el número de evaluaciones de la función de distancia observadas entre ellos, muestran que KVP necesita el doble que UPI, salvo en U14 y en NASA donde necesita aproximadamente un 50 y un 36 % más. Esto es indicativo de las diferencias de criterios seguidos en ambas propuestas, como por ejemplo la importancia dada en UPI a la generación de pivotes que realmente sean efectivos. Si la mejora obtenida con UPI en el número de evaluaciones de la función de distancia es notoria, en el caso del espacio son abrumadoras, ya que KVP ocupa del orden de cinco veces el espacio que necesita UPI. Con todos estos datos se concluye que UPI representa un importante salto cualitativo con respecto a KVP.

Quedan por analizar los resultados de UPI frente a una alternativa representativa de la eficiencia que obtienen los métodos basados en particiones, como es el caso de Lista de Clusters. En términos de evaluaciones de la función de distancia, UPI mejora a Lista de Clusters en todas las colecciones de vectores, ya que este último necesita, desde unas cuantas comparaciones más en el caso de NASA, hasta casi el doble para U08 y U10. Solo se presenta una excepción a este comportamiento en la colección de palabras ENGLISH, en la que UPI necesita del orden de un 12 % más de evaluaciones. En lo que respecta al tamaño del índice, UPI mejora sensiblemente a Lista de Clusters, ya que este genera índices que ocupan entre un 125 y un 141 % con respecto al 100 % de UPI. Estos datos resaltan la eficiencia de UPI y justifican que se considere como una alternativa competitiva frente a los métodos basados en particiones, ya que casi siempre consigue una reducción del número de evaluaciones de la función de distancia, y además obtiene mejores resultados en el tamaño del índice, precisamente uno de los puntos fuertes de los métodos basados en particiones.

4.9. Resumen

En este capítulo se ha introducido un nuevo método de búsqueda por similitud basado en pivotes, cuyo objetivo inicial es reducir el espacio asociado al tamaño del índice, un problema que presentan habitualmente los métodos basados en pivotes, que dificulta y a veces impide su aplicación a casos reales en los que se manipulan colecciones de datos muy voluminosas, porque estos métodos se basan en almacenar en el índice las distancias de los pivotes al resto de objetos de la base datos, para después utilizar esas distancias en el proceso de búsqueda, con el objetivo de descartar objetos sin tener que calcular directamente su distancia al objeto de consulta.

En muchos casos la comprobación de la eficiencia de los métodos basados en pivotes, se centra en considerar únicamente el número de evaluaciones de la función de distancia, desestimando el impacto que el tamaño del índice puede tener por considerarlo mucho menor que el primero. Pero el tamaño del índice tiene incidencia en otros factores que forman parte del coste total de las búsquedas, como el tiempo de entrada/salida y el tiempo resultante de procesar el índice. Se ha partido de ese hecho para presentar los puntos débiles de los métodos basados en pivotes, en concreto los requerimientos de espacio y el criterio de selección de los pivotes, y se han mencionado soluciones a esos problemas. Seguidamente se han estudiado los factores que contribuyen a incrementar la eficacia de los pivotes, orientándolos a su capacidad para descartar cada uno de los objetos de la base de datos, y no la totalidad de los mismos, resultando que el pivote más cercano y el más lejano a un objeto son los que presentan mejores condiciones para descartarlo. Posteriormente se ha realizado un análisis de la eficiencia de los pivotes, estudiando el efecto de la elección del conjunto inicial de pivotes y de la selección de los más prometedores para cada objeto, además de intentar que el índice ocupase un espacio muy reducido. El resultado ha proporcionado unos parámetros rigurosos para elegir los pivotes en cada colección.

En este punto se ha propuesto un nuevo método basado en pivotes, que almacena para cada objeto únicamente la distancia a un pivote, que es el más cercano o el más lejano, según los valores de los parámetros determinados para la colección, con lo que se está eligiendo un pivote que tiene realmente muchas posibilidades de provocar el descarte del objeto. En todo momento se ha considerado el problema del número y de la distribución de los pivotes seleccionados, por lo que se ha utilizado un criterio contrastado como SSS para el que se ha comprobado su viabilidad en el escenario propuesto. El método combina las estrategias de reducción de rango y de ámbito, ya que almacena solo la distancia más prometedora para cada objeto y para ello utiliza la mitad de la precisión.

Se ha considerado la posibilidad de almacenar en el índice las distancias con menor precisión, lo que ha resultado adecuado, y se ha descrito la manera de realizar la búsqueda por rango. Se ha tratado también el procedimiento a seguir en la búsqueda kNN, incorporando modificaciones que mejoran su eficiencia.

El método se ha contrastado experimentalmente para la búsqueda por rango sobre algunas de las colecciones, tanto reales como sintéticas, pertenecientes a la *Metric Spaces Library*, obteniendo unos resultados que muestran que supera ampliamente a otros métodos basados en pivotes en la reducción del espacio ocupado por el índice, y que además presenta un número de evaluaciones de la función de distancia aceptable. Se ha comprobado también que es una alternativa a los métodos basados en particiones, ya que supera a un método representativo como es Lista de Clusters, tanto es espacio como en evaluaciones de la función de distancia, en casi todas las colecciones analizadas. Este comportamiento es inusual en otros métodos basados en pivotes.

Capítulo 5

Reducción de cluster

En este capítulo se presenta una nueva estrategia que mejora el rendimiento de las búsquedas por similitud en los métodos basados en particiones que usan como criterio de descarte el radio de cobertura, gracias a que reduce el número de evaluaciones de la función de distancia al explorar los clusters que no pueden descartarse, esto es, que reduce el valor de la complejidad externa.

En la sección 5.1 se plantean los problemas que dieron origen a considerar un nuevo enfoque, en la siguiente sección se presenta la nueva estrategia, en la sección 5.3 se exponen los algoritmos de búsqueda por rango y de búsqueda kNN, en la sección 5.4 se presentan los resultados de la evaluación experimental realizada, en la siguiente sección se analizan esos resultados, en la sección 5.6 se discuten las posibilidades dinámicas y finalmente se hace un resumen del capítulo.

5.1. Motivación

Como se ha visto en la sección 2.4, los factores que más influyen en la eficiencia de las búsquedas por similitud, se centran en el número de evaluaciones de la función de distancia y en el tamaño de los índices o estructuras auxiliares utilizadas para acelerar las búsquedas.

Los métodos basados en particiones superan a los basados en pivotes en que requieren un menor espacio para sus índices y en su mejor comportamiento en espacios de grandes dimensiones, pero el número de evaluaciones de la función de distancia que originan es mayor, como se ha plasmado en la figura 2.26.

Los métodos basados en particiones o en clustering, dividen el espacio en zonas, denominadas en la literatura anglosajona "clusters", siguiendo normalmente, o bien el criterio de las zonas esféricas, o bien el de los hiperplanos.

Dada una consulta, estos métodos comparan el objeto de búsqueda q con los centros de cada cluster, de forma que, usando las propiedades de los espacios métricos y las características de los clusters almacenadas en el índice, se intenta descartar el mayor número posible de clusters, sin comparar sus objetos con q, pudiendo hacerlo si se garantiza que ninguno de los elementos de un cluster cumple el criterio de búsqueda. Se denomina complejidad interna al número de evaluaciones de la función de distancia necesarias para comparar el objeto de búsqueda con los centros. Si un cluster no puede descartarse, se calcula la distancia de cada uno de sus elementos a q, en un proceso de exploración exhaustiva. La complejidad externa es el número de comparaciones de la función de distancia que se realizan en este proceso de exploración de los clusters no descartados.

Reducir el número de evaluaciones de la función de distancia es un importante objetivo en los métodos basados en particiones, porque incide en el factor que supone la debilidad crucial de estos métodos. Cualquier estrategia que lleve a reducir ese número, sin ocasionar variaciones importantes en los restantes factores involucrados en la eficiencia, puede suponer una aportación interesante.

5.2. Reducción de cluster

La técnica de *Reducción de Cluster* o *Cluster Reduction (CR)* es una estrategia aplicable a los métodos basados en particiones que utilizan el radio de cobertura como criterio de descarte de los clusters. Consiste en dividir cada cluster en varias regiones, determinadas por las distancias del centro a una serie de objetos del propio cluster, actuando dichas distancias como radios de cobertura interiores, lo que permite que durante el proceso de búsqueda se reduzca progresivamente el cluster en función de esas regiones, de forma que sobre cada cluster no descartado, se realice la comparación de distancias solo en sus regiones imprescindibles, y no de manera exhaustiva en la totalidad del cluster. De esta forma se origina un número menor de evaluaciones de la función de distancia, con lo que se reduce la complejidad externa. El índice necesita almacenar la referencia de qué objetos determinan las regiones, las distancias de esos objetos al centro del cluster y a qué región pertenecen los restantes objetos. De esta forma se incrementa el tamaño del índice, pero este efecto puede ser muy reducido, porque por una parte, se mantienen los requerimientos de espacio en $O(n)$, y por otra, la evaluación experimental muestra que, con un pequeño incremento en el tamaño del índice, se obtiene una mejora significativa del rendimiento de las búsquedas.

En los métodos basados en particiones, el proceso inicial de generación de clusters comienza con el particionado del espacio utilizando un método determinado. Con la estrategia de Reducción de Cluster se puede usar tanto el particionado mediante zonas esféricas como utilizando hiperplanos, ya que solo condiciona el criterio de descarte de clusters utilizado y no el del particionado elegido. Por ejemplo, puede particionarse el espacio usando hiperplanos y después aplicar Reducción de Cluster para realizar las búsquedas, por lo que en este punto se utilizará el radio de cobertura para descartar los clusters, como se ha visto en BST (página 77).

Una vez realizado el particionado, se tiene un número m de clusters $\{C_1, ..., C_m\}$ con sus respectivos centros u objetos de referencia $\{c_1, ..., c_m\}$. Se tienen también los radios de cobertura de cada cluster $\{r_{c_1}, ..., r_{c_m}\}$, que se utilizaron en el proceso de particionado, si este se realizó mediante zonas esféricas, pero que se utilizarán en el proceso de búsqueda al aplicar Reducción de Cluster.

La estrategia de Reducción de Cluster divide ahora cada cluster en β regiones, determinadas por las distancias de su centro a $\beta - 1$ objetos, de forma que estas distancias actúan como radios de cobertura interiores de cada cluster. La selección de las regiones, y por tanto de los objetos que las determinan, se realiza intentando que todas ellas tengan el mismo número de elementos, salvo en los casos en los que, debido a que el número de objetos del cluster no es divisible por el número de regiones, es necesario ajustar el número de objetos en una de las regiones.

Si en la selección de las regiones se usan otros criterios, como por ejemplo que todas las regiones contiguas tuviesen la misma diferencia en las longitudes de sus radios de cobertura, lo que llevaría a que las regiones tuviesen lo que podríamos denominar "la misma anchura", no se garantiza, en general, una división efectiva de los objetos del cluster, porque la distribución de los objetos del cluster no es uniforme con respecto a su centro [Chávez y Navarro, 2005]. En un caso límite podría ocurrir que todos los objetos del cluster se concentrasen en una región determinada, por ejemplo en la más alejada del centro, con lo que la división en regiones realmente no afectaría a los objetos del cluster.

Se tiene así cada cluster dividido en una serie de regiones a las que referenciamos según su lejanía al centro, de forma que indicamos que la primera es la más alejada del centro, la segunda es la siguiente y así sucesivamente.

La figura 5.1 muestra un cluster particionado mediante zonas esféricas, en su estado inicial, tal como lo tratan los métodos de clustering. En la zona superior de la figura 5.2 se representa el mismo cluster al que se le ha aplicado Reducción de Cluster, en este caso dividiéndolo en $\beta = 5$ regiones mediante la elección de $\beta - 1 = 4$ objetos. El centro del cluster es c, su radio de cobertura es r_c y el número de objetos del cluster, sin contar su centro, es 20. Se ha dividido en cinco regiones, delimitadas con un trazo discontinuo, para lo que se han considerado cuatro objetos del cluster, de forma que cada región acaba conteniendo también a

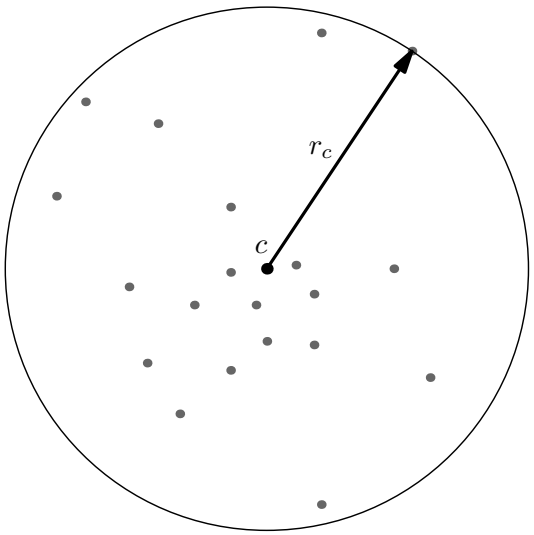
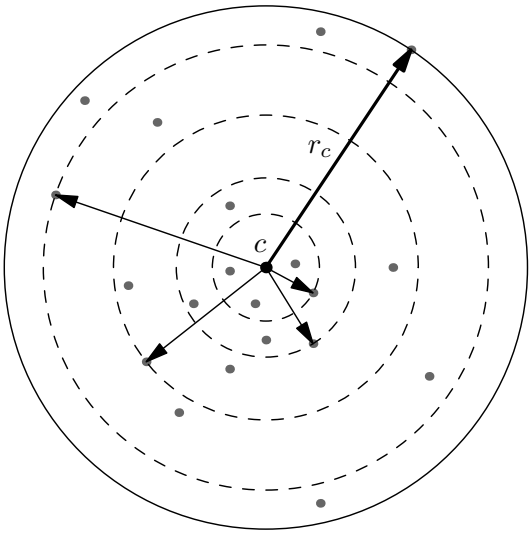


Figura 5.1: Estado inicial de un cluster particionado mediante zonas esféricas.

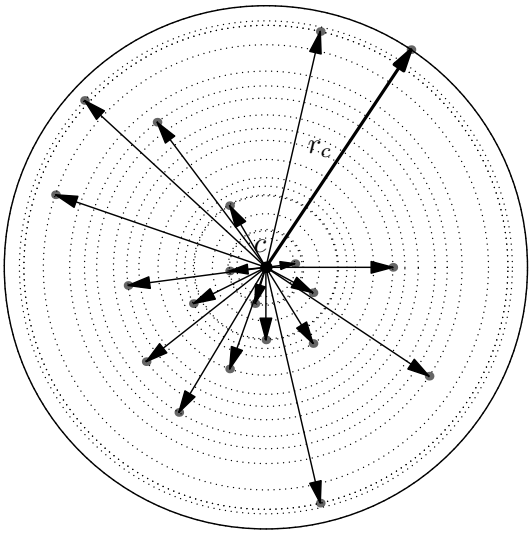
cuatro objetos y se mantiene así un número de objetos similar en cada región, con la salvedad del propio centro del cluster. Las regiones están determinadas por las distancias del centro del cluster a los cuatro objetos interiores, indicadas en la figura mediante flechas de trazo más fino que la correspondiente al radio de cobertura.

Las distancias del centro de un cluster a los objetos que definen sus regiones, son también cotas de las distancias del centro del cluster a los objetos contenidos en la región que delimitan. Por ello, la definición de las regiones de la manera propuesta, hace que cada centro juegue el papel de pivote de los objetos de su cluster, no solo de los objetos que determinan las regiones, porque para ellos se almacenan sus distancias al centro y esto hace que trivialmente dicho centro se comporte como pivote de esos objetos, sino que además permite establecer que para cada objeto del cluster, su distancia al centro está en un rango determinado por las cotas de la región a la que pertenece. El centro actúa así como pivote de los objetos interiores a las regiones, no de manera exacta sino aproximada, ya que puede determinarse entre qué distancias se encuentra un objeto, en un ejemplo de la técnica denominada degradación del rango mencionada en la sección 4.2.1.

La principal diferencia de Reducción de Cluster respecto a las propuestas existentes es que divide cada cluster en regiones concéntricas con respecto a su centro, con el objetivo de que almacenando las distancias a los objetos que



(a) División en cinco regiones ($\beta = 5$)



(b) División en tantas regiones como objetos ($\beta = all$)

Figura 5.2: División de un cluster en regiones.

determinan las regiones, se pueda realizar una progresiva reducción del cluster durante el proceso de búsqueda, que permita descartar regiones sin comparar sus objetos con el objeto de búsqueda y evite así la exploración exhaustiva de todos los objetos del cluster, esto es, la comprobación de si cada objeto del cluster cumple el criterio de búsqueda mediante el cálculo de su distancia al objeto de búsqueda.

En el índice asociado a un método de clustering que utiliza el radio de cobertura como criterio de descarte, se almacena información referente a los clusters, como su centro, su radio de cobertura y los objetos que pertenecen a cada cluster. Para aplicar la estrategia de Reducción de Cluster a un método de ese tipo, se necesita que el índice incorpore, para cada cluster, lo siguiente:

- Las referencias de los objetos que determinan las regiones de cada cluster.
- Las distancias de esos objetos al centro del cluster.
- A qué región pertenecen los restantes objetos del cluster.

El índice puede implementarse usando diversas estructuras de datos, ya que al no realizarse un particionado recursivo del espacio no se requiere que sea arbóreo. Una alternativa que necesita un espacio mínimo consiste en utilizar una estructura con dos arrays, uno para almacenar las distancias de los objetos que definen las regiones al centro del cluster, y otro que contiene los indicadores de todos los objetos, ordenados por su pertenencia a las distintas regiones y comenzando por la más alejada del centro del cluster.

El número de regiones que pueden definirse en cada cluster va desde un mínimo de dos, que sería el caso de usar únicamente la distancia a un objeto interior, hasta un máximo determinado por el número de objetos en el cluster menos una unidad, que se daría si se utilizan todos los objetos del cluster menos su centro, como se muestra en la zona inferior de la figura 5.2. En el primer caso se podría descartar la mitad de los objetos del cluster sin compararlos con el objeto de búsqueda; en el segundo, al almacenar las distancias de cada objeto al centro, se podrían descartar todos los objetos restantes a partir de uno dado.

Cuantas más regiones se definen, mayores son las posibilidades de descartar más objetos, haciendo que el número de evaluaciones de la función de distancia se reduzca. Pero al almacenar en el índice las distancias del centro a cada objeto que determina una región, a medida que aumenta el número de regiones, se incrementa también el tamaño del índice, por lo que es necesario establecer un equilibrio entre el coste de la búsqueda y los requerimientos del espacio ocupado por el índice.

La expectativa que fundamenta la hipótesis planteada en este capítulo, es que la división de los clusters en un número reducido de regiones de la forma que se realiza en Reducción de Cluster, reduce significativamente el número de evaluaciones de la función de distancia, a cambio de un pequeño incremento en el tamaño del índice.

5.3. Búsqueda por similitud con Reducción de Cluster

En los métodos basados en particiones, ante una consulta determinada, se intenta descartar el máximo número de clusters realizando el menor número de comparaciones de la función de distancia. Para ello se calcula la distancia del objeto de búsqueda a los centros de los clusters y así se comprueba si dichos clusters pueden contener objetos que cumplen el criterio de búsqueda. Si un cluster no resulta descartado se necesita realizar una comprobación adicional, que consiste en calcular la distancia existente entre cada uno de los objetos que contiene el cluster y el objeto de búsqueda, en un proceso de exploración exhaustiva del cluster.

El método de Reducción de Cluster trata de evitar la exploración exhaustiva de los clusters que no han sido descartados, mediante su división en regiones, de forma que las búsquedas se circunscriban al número de regiones imprescindibles, con lo que el cálculo de las distancias se realiza únicamente para los objetos de esas regiones y no para la totalidad de los objetos del cluster. Esta reducción del número de evaluaciones de la función de distancia incide sobre uno de los factores de la complejidad de las búsquedas, la denominada complejidad externa.

El proceso guarda un parecido con el descarte de clusters que se origina en los métodos de clustering, ya que al igual que en ellos se descartan los clusters para los que se garantiza que no poseen objetos que cumplen el criterio de búsqueda, aquí se descartan regiones que están en esa misma situación. También recuerda a los métodos de pivotes porque los centros de los clusters actúan como pivotes de los objetos que determinan las regiones, lo que además de permitir comprobar de manera inmediata si uno de esos objetos cumple el criterio de búsqueda, hace posible que se detecten las regiones que contienen objetos que pueden cumplir el criterio, sin realizar nuevos cálculos de distancias, gracias a que las distancias del centro a los objetos que determinan las regiones establecen unas cotas para las distancias a los restantes objetos de las regiones.

En los apartados siguientes se explican detalladamente los algoritmos de búsqueda por rango y búsqueda kNN aplicando Reducción de Cluster al método de Lista de Clusters, aunque podría aplicarse a cualquier otro método basado en particiones que use el radio de cobertura en el criterio de descarte de los clusters.

5.3.1. Búsqueda por rango

En la búsqueda por rango mediante Reducción de Cluster se procede inicialmente como en otros métodos de clustering existentes que utilizan el criterio de descarte basado en el radio de cobertura, esto es, se determina qué clusters pueden

descartarse sin calcular las distancias a sus objetos. Para ello, dada una consulta $R(q, r)$, se calculan las distancias del objeto a buscar q a los diferentes centros de los clusters. Dado que cada centro de cluster y su radio de cobertura definen un círculo (c, r_c), y que el objeto a buscar y el rango de la consulta determinan el círculo de búsqueda (q, r), una vez calculada la distancia del centro al objeto de búsqueda, se puede comprobar si ambos círculos tienen intersección vacía, con lo que el cluster podría descartarse.

La estrategia propuesta en este capítulo se diferencia de la utilizada en los métodos existentes, en la manera de actuar en el caso de no poder descartar un cluster. Se pueden dar las tres posibilidades siguientes:

1. Si $d(q, c) - r_c > r$, entonces el círculo definido por el cluster, (c, r_c), no interseca al círculo de búsqueda, (q, r), por lo que ninguno de sus objetos puede cumplir el criterio de búsqueda y, por tanto, la totalidad del cluster se descarta.
2. Si $d(q, c) - r_c \leq r$ pero no ocurre que $d(q, c) + r \leq r_c$, entonces el círculo definido por el cluster interseca al círculo de búsqueda, por lo que algunos de los objetos del primero pueden cumplir el criterio de búsqueda. En este caso, los métodos existentes exploran la totalidad del cluster comparando q con cada uno de sus objetos. Nuestra estrategia realiza una división del cluster en regiones y aprovecha ahora esa división para efectuar también una comparación por regiones. Se realiza una reducción del cluster mediante las regiones definidas por los radios de cobertura interiores, empezando por las regiones más alejadas del centro, de forma que pueden descartarse todas las regiones restantes a partir de una dada, si la siguiente región no interseca al círculo de búsqueda.
3. Si $d(q, c) - r_c \leq r$ y además $d(q, c) + r \leq r_c$, entonces el círculo de búsqueda no solo interseca al círculo definido por el cluster, sino que está contenido en él. La comprobación de las distancias de q a los objetos del cluster, se ciñe a las regiones que intersecan al círculo de búsqueda, por lo que en este caso pueden resultar eliminadas tanto las regiones más alejadas del centro, como las más cercanas a él. Además en este caso la búsqueda finaliza, porque los objetos de los otros clusters no pueden cumplir el criterio de búsqueda.

La figura 5.3 muestra tres ejemplos de búsqueda por rango con Reducción de Cluster. Se tiene un cluster de centro c y radio de cobertura r_c, con veinte objetos en total, además del centro, que se ha dividido en cinco regiones determinadas por las distancias del centro a cuatro objetos interiores, representadas mediante flechas de trazo más fino. Las tres consultas con sus respectivos objetos a buscar y sus rangos, corresponden, respectivamente, a cada una de las diferentes posibilidades mencionadas:

1. La búsqueda $R(q, r)$ es un ejemplo de la primera posibilidad porque ocurre que $d(q, c) - r_c > r$, con lo que el círculo de búsqueda tiene intersección vacía con el círculo definido por el cluster, y esto permite asegurar que ningún objeto del cluster cumple el criterio de búsqueda, por lo que el cluster puede descartarse completamente sin calcular las distancias de sus objetos al objeto de búsqueda q.
2. La búsqueda $R(q', r')$ representa la segunda posibilidad, ya que se tiene que $d(q', c) - r_c \leq r'$ y por ello el círculo de búsqueda interseca de manera no vacía al círculo definido por el cluster. El procedimiento a seguir consiste en recorrer las diferentes regiones del cluster empezando por la más alejada del centro, calcular las distancias de sus objetos al objeto de búsqueda, reducir después el cluster eliminando la región ya explorada y reiterar la comprobación hasta que la intersección del cluster reducido con el círculo de búsqueda sea vacía. Cuando eso ocurra, el resto de regiones del cluster puede descartarse, puesto que está garantizado que sus objetos no cumplen el criterio de búsqueda. En la figura se observa que la intersección es no vacía para la región más alejada del centro, por lo que se calculan las distancias de sus objetos a q', para después reducir el cluster eliminando dicha región. Una vez reducido el cluster, su intersección con el círculo de búsqueda es vacía, por lo que ya no es necesario comprobar las cuatro regiones restantes, con lo que la consulta se resuelve realizando únicamente el 20 % de las comprobaciones de la función de distancia, con respecto al total de objetos del cluster.
3. La búsqueda $R(q'', r'')$ es un ejemplo del caso particular en el que el círculo de búsqueda está contenido en la zona delimitada por el cluster, porque $d(q'', c) + r'' \leq r_c$. La comprobación de qué regiones tienen intersección no vacía con el cluster se realiza mediante el proceso de reducción del mismo, sin necesidad de calcular la función de distancia, empezando por la región más alejada del centro, que resulta descartada y se reduce el cluster. A continuación para las dos regiones siguientes hay que calcular la distancia de sus objetos a q'', porque intersecan al círculo de búsqueda. La cuarta región vuelve a originar una intersección vacía, por lo que, además de descartarla, provoca que se desechen las restantes regiones, en este caso la quinta y última. La consulta se ha resuelto comparando q'' con un 40 % de los objetos del cluster.

   El hecho de que el círculo de búsqueda esté contenido en el círculo definido por el cluster, como en este caso, tiene una consecuencia adicional, y es que no es necesario comprobar los restantes clusters, porque todos los objetos que cumplen el criterio de búsqueda están ya en el actual, por lo que los demás clusters pueden descartarse.

La similitud con los métodos basados en pivotes viene dada porque el centro c juega el papel de pivote de los objetos de su cluster. Recordando la zona de exclusión para una búsqueda por rango, definida por un pivote determinado, en

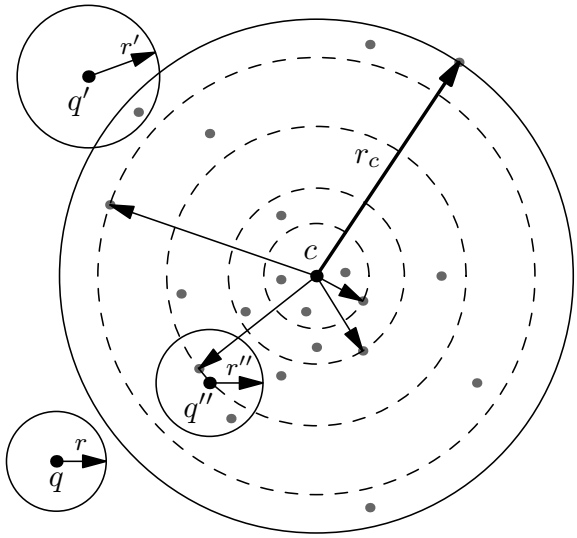


Figura 5.3: Ejemplos de búsquedas por rango con Reducción de Cluster.

la que no se podían descartar sus objetos y representada en la figura 2.12 de la página 36, se tiene que una vez calculada la distancia del centro a un objeto de búsqueda, se puede determinar una zona de exclusión en función del centro. Para cada consulta dada, las regiones del cluster que intersecan a la zona de exclusión son las únicas que deben explorarse. En la figura 5.4 se observa que para la consulta $R(q, r)$ la región de exclusión está delimitada por el trazo de puntos en azul, que está contenida en la región más alejada del centro, por lo que las restantes pueden descartarse. Análogamente ocurre para la consulta $R(q', r')$, cuya zona de exclusión, representada ahora en color naranja, está contenida en las regiones segunda y tercera, que son las únicas que es necesario explorar, pudiendo descartar las restantes regiones y reducir así el número de evaluaciones de la función de distancia.

5.3.2. Búsqueda kNN

Los métodos de clustering que basan su criterio de descarte en el radio de cobertura, resuelven de manera general una búsqueda $kNN(q)$ utilizando las distancias del objeto de búsqueda a los centros y los radios de cobertura, para determinar, por una parte, una primera lista de candidatos a k vecinos más cercanos formada por los centros más próximos, y por otra, el cluster más cercano al objeto

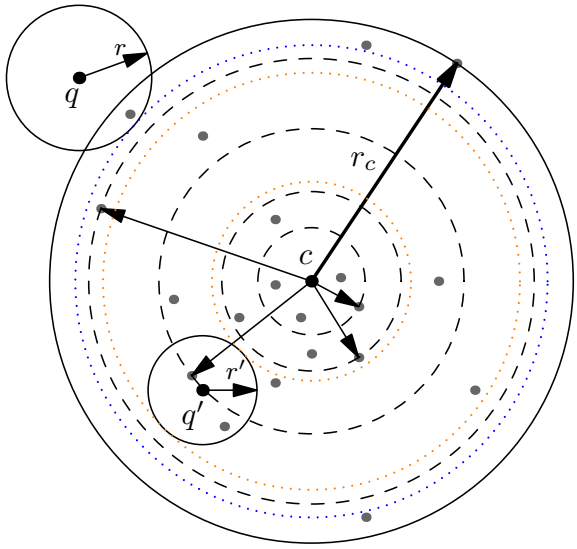


Figura 5.4: Similitud del proceso de determinar las regiones a explorar con el de pivotes.

de búsqueda, denominado frecuentemente *cluster más prometedor*. Sobre ese cluster se realiza una exploración exhaustiva, calculando para cada uno de sus objetos su distancia a q y actualizando la lista de candidatos. El proceso continúa actuando sobre los siguientes clusters más cercanos a q y finaliza cuando el siguiente cluster más cercano está más lejos de q que el actual k-ésimo vecino más cercano.

La búsqueda $kNN(q)$ usando Reducción de Cluster se diferencia en que, una vez determinado el cluster más prometedor, no se compara el objeto de búsqueda q con todos los objetos del cluster, sino que se compara solo con aquellos que pertenecen a la región del cluster más cercana a q. El cluster sobre el que se ha actuado se sigue considerando en el resto del proceso de búsqueda, pero ahora sin la región ya explorada. La búsqueda continúa explorando el siguiente cluster más prometedor, hasta que la siguiente región más cercana al objeto de búsqueda se encuentre más lejana que el actual k-vecino más cercano a q, momento en el que la búsqueda finaliza.

Las búsquedas se restringen así a las regiones más cercanas a q y no abarcan la totalidad de los clusters, al igual que ocurría en las búsquedas por rango. Se puede decir que se produce también un cambio de óptica, pasando de buscar en el cluster más prometedor a hacerlo sobre la región más prometedora, porque después de

determinar el cluster más prometedor, se actúa solo sobre la región más cercana al objeto a buscar, se actualiza la lista de candidatos y se reduce el cluster eliminando esa región, para reiterar el procedimiento considerando ahora el cluster reducido. Esta estrategia permite actuar directamente sobre las regiones imprescindibles e ir reduciendo los clusters, para buscar la siguiente región que se encuentre más próxima al objeto de búsqueda, sea en el mismo cluster o en cualquier otro. Podría ocurrir que primero se explorase una región de un cluster y una vez elimina esa región, la siguiente a explorar resultase otra región de otro cluster, provocando un efecto de optimización al explorar las regiones más cercanas al objeto de búsqueda, independientemente del cluster en el que se encuentran.

Para determinar cuál es el cluster más prometedor, se tiene en cuenta que dado un cluster $\mathcal{C}$ de centro c y radio de cobertura r_c, la expresión $d(q,c) - r_c$ es una cota inferior para la distancia de q a cualquier objeto de $\mathcal{C}$, esto es,

$$\forall\, x \in \mathcal{C}, \quad d(q,x) \geq d(q,c) - r_c \tag{5.1}$$

por lo que dados dos clusters $\mathcal{C}_i$ y $\mathcal{C}_j$, el cluster $\mathcal{C}_i$ es más prometedor que $\mathcal{C}_j$ si:

$$d(q,c_i) - r_{c_i} < d(q,c_j) - r_{c_j} \tag{5.2}$$

Se representa así de manera formal la idea de que $\mathcal{C}_i$ está más próximo a q. Si el objeto de búsqueda se encontrase dentro de los límites del círculo definido por el cluster $\mathcal{C}_i$, el lado izquierdo de la inecuación 5.2 sería negativo, haciendo que esta se cumpliese de manera trivial.

El objeto de búsqueda se compara con todos los centros de los clusters originando una cola de prioridad en la que se colocan primero los clusters más cercanos a q. En cada paso se extrae de la cola el cluster más prometedor y se explora su región más cercana al objeto de búsqueda, actualizando la lista de candidatos, para luego eliminarla del cluster, provocando su reducción, y después introducirlo en la cola con una nueva prioridad, determinada en función de su actual distancia a q. El proceso se repite hasta que la cota inferior de la distancia de q al siguiente cluster más prometedor es mayor que la distancia al actual k-ésimo candidato a vecino más cercano.

El algoritmo 1 muestra el pseudocódigo de la búsqueda $kNN(q)$ usando Reducción de Cluster. Se usan dos colas de prioridad, *clustersQueue* que almacena los clusters según su criterio de proximidad al objeto de búsqueda o cómo de prometedores son en relación la búsqueda, y *neighborsQueue* para mantener la lista de candidatos a k vecinos más cercanos a q. La función *Reduce* realiza la exploración de la región de C_i más cercana a q, actualiza la cola de prioridad de los candidatos a vecinos más cercanos y elimina del cluster la región que acaba de explorar.

Algoritmo 1: Pseudocódigo de la búsqueda kNN con Reducción de Cluster.

```
1  kNNSearch(q)
2  foreach cluster C_i do
3  |   d_i ← d(q, c_i)
4  |   Insert(clustersQueue, C_i, d_i − r_{c_i})
5  end
6  C_i ← pull(clustersQueue)
7  while d(q, c_i) − r_{c_i} < radius(neighborsQueue) do
8  |   Reduce(q, C_i, neighborsQueue)
9  |   Insert(clustersQueue, C_i, d_i − r_{c_i})
10 |   C_i ← pull(clustersQueue)
11 end
```

La figura 5.5 refleja un ejemplo de búsqueda 2NN sobre un objeto de búsqueda q en un espacio particionado en dos clusters $\mathcal{C}_1$ y $\mathcal{C}_2$, de centros c_1 y c_2 respectivamente. El proceso se inicia calculando las distancias de q a los centros de los clusters, de forma que los centros pasan a formar parte de la lista de candidatos a 2-vecinos más cercanos. Dado que la cota inferior de las distancias de q a cualquier objeto de $\mathcal{C}_1$, representada por d_1, es menor que la correspondiente al cluster $\mathcal{C}_2$, representada por d_2, se tiene que el cluster $\mathcal{C}_1$ es el más prometedor. La estrategia de Reducción de Cluster no compara ahora q con la totalidad de los objetos de $\mathcal{C}_1$, como hacen los métodos existentes, sino solo con los de la región de $\mathcal{C}_1$ más próxima a q, que es la más alejada del centro del cluster. Después de la exploración de dicha región, se actualiza la lista de candidatos a vecinos más cercanos, reemplazando en ella los dos centros de los clusters por los dos objetos encontrados en la región. A continuación se reduce el cluster $\mathcal{C}_1$ eliminando la región que ya ha sido explorada, pasando a ser la nueva cota inferior de las distancias de q a cualquier objeto del cluster reducido, la distancia representada por d_3.

El proceso de búsqueda prosigue determinando de nuevo el cluster más prometedor, y como d_2 es menor que d_3, resulta ser $\mathcal{C}_2$, por lo que se repite el procedimiento de exploración de su región más cercana a q, que es la más alejada de su centro. Los dos objetos que posee pasan también a ser los nuevos candidatos a vecinos más cercanos y después se reduce el cluster al eliminar esa región. El siguiente cluster más prometedor lo determina el hecho de que la distancia d_4 es menor que d_3, por lo que se explora la segunda región de $\mathcal{C}_2$, pero ocurre que los objetos que se encuentran en ella no modifican la lista de candidatos, porque están más alejados de q que los dos candidatos que actualmente hay en la lista. Las siguientes regiones a explorar las determinan las distancias d_3 y d_5, pero al ser

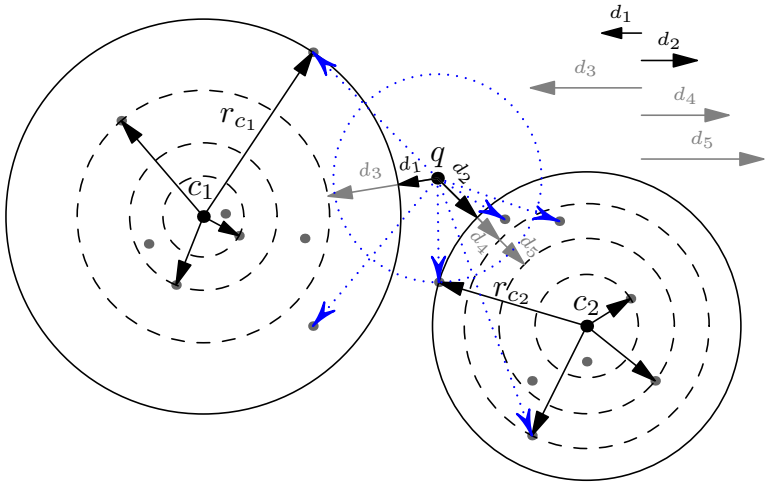


Figura 5.5: Ejemplo de búsqueda kNN con Reducción de Cluster.

mayores que la distancia de q al segundo candidato a vecino más cercano, que está representada en la figura por la circunferencia discontinua, se tiene que cualquier objeto de las regiones asociadas a esas distancias estaría a una distancia mayor que la existente a ese candidato, con lo que se garantiza que no se necesita explorar ninguna región más y la búsqueda finaliza.

En este ejemplo la búsqueda se ha resuelto realizando la exploración, esto es, el cálculo de la función de distancia, en únicamente tres de las ocho regiones en las que se habían dividido los dos clusters, con lo que se ha reducido a solo un 37,5 % del total de posibles comparaciones. Además se ha explorado primero una región de un cluster y después se ha saltado a otra del otro cluster, reflejando el carácter optimizador de esta estrategia, ya que en cada iteración del proceso, se centra en la región más adecuada para realizar la búsqueda, esto es, en la más cercana al objeto a buscar, independientemente del cluster al que pertenezca.

Casos particulares de la búsqueda kNN

Si el objeto a buscar q, se encuentra dentro del círculo formado por el centro del cluster y el radio de cobertura, se procede de manera similar al ejemplo anterior. La figura 5.6 muestra un ejemplo de esta situación, ya que q pertenece a la región más alejada del centro del cluster de la izquierda. Para ese cluster ocurre que

$d(q, c_1) - r_{c_1} \leq 0$, con lo que se actúa primero sobre él, concretamente sobre su región más próxima a q. En esa región se calculan las distancias de q a los dos objetos que contiene y se actualiza la lista de candidatos a vecinos más cercanos. Uno de los objetos de la región, el más cercano a q, sustituye al centro c_2 en esa lista de candidatos. Finalmente, se elimina la región ya explorada. Al reducir el cluster eliminando su región más exterior ocurre que el cluster reducido ya no contiene al objeto de búsqueda, por lo que la búsqueda continúa como en el ejemplo anterior.

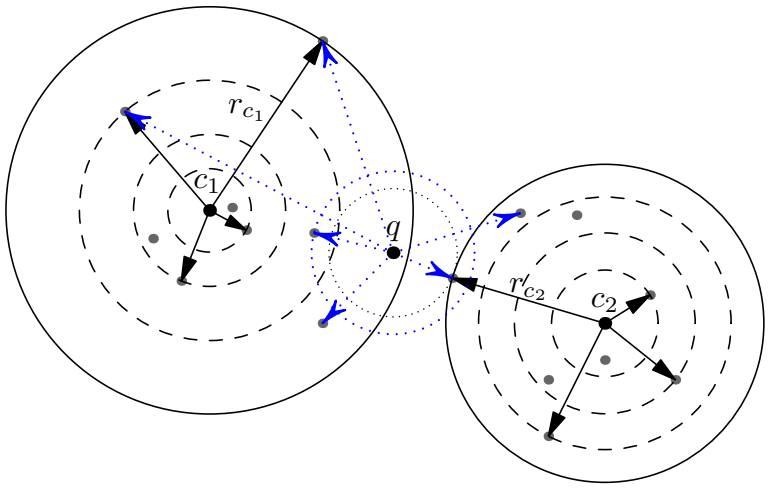


Figura 5.6: Segundo ejemplo de búsqueda kNN con Reducción de Cluster.

El ejemplo tratado clarifica también la manera de proceder del algoritmo, que siempre actúa sobre la región más cercana al objeto de búsqueda, la explora y la elimina del cluster al que pertenece, para reiterar el procedimiento ahora sobre la nueva región más cercana al objeto de búsqueda, pero considerando ya el cluster reducido.

Resulta también interesante mostrar el comportamiento del algoritmo si el objeto de búsqueda está dentro de los límites de una región interior. Esta situación se representa en la figura 5.7, en la que q está contenido en la segunda región del cluster de centro c_1 que está más alejada del propio centro. En primer lugar ocurre que $d(q, c_1) - r_{c_1} \leq 0$, por lo que se actúa sobre ese cluster.

Siendo β el número de regiones en las que se dividen los cluster y denotando con $r_{c_{11}}$ al radio de cobertura del cluster de centro c_1, esto es, $r_{c_{11}} = r_{c_1}$ y

con $r_{c_{12}}, ..., r_{c_{1\beta}}$ a sus radios de cobertura interiores en orden decreciente, ocurre que, una vez calculada la distancia del objeto de búsqueda al centro del cluster y sin realizar nuevos cálculos de la función de distancia, es inmediato determinar la región a la que pertenece q, ya que es la que cumple una de las dos condiciones siguientes, o bien q pertenece a la i-ésima región si:

$$r_{c_{1i}} \geq d(q,c) > r_{c_{1i+1}} \quad \text{para algún } i = 1, ..., \beta - 1 \tag{5.3}$$

o bien q pertenece a la β-ésima región, la más cercana al centro, si:

$$r_{c_{1\beta}} \geq d(q,c) \tag{5.4}$$

Primero se explora la región a la que pertenece el objeto de búsqueda, que en la figura 5.7 es la segunda más alejada del centro del cluster de la izquierda, se actualiza la lista de candidatos a vecinos más cercanos y después se elimina la región, reduciendo el cluster. A continuación se determina la siguiente región más cercana a q, teniendo en cuenta que al cluster de centro c_1 se le ha eliminado la segunda región más alejada del centro del cluster, por lo que la siguiente más cercana a q es la más alejada del centro que pertenece al mismo cluster, en la que uno de sus objetos pasa a la lista de candidatos a vecinos más cercanos. Después de eliminar también esa región, aún hay otra región que puede tener objetos que están más cerca de q que el actual segundo vecino más cercano, y es la que ahora está más alejada del centro del cluster, la que inicialmente era la tercera más alejada del centro, por lo que es necesario explorarla. Después de hacerlo, resulta que no se modifica la lista de candidatos, y la región se elimina del cluster. Las siguientes regiones, tanto de ese cluster como del otro, están ya a una distancia mayor de q que el actual segundo vecino más cercano, por lo que la búsqueda finaliza, pasando los objetos de la lista de candidatos a ser los 2-vecinos más cercanos a q.

5.4. Resultados experimentales

En esta sección se presenta el planteamiento de la evaluación experimental de los algoritmos de Reducción de Cluster, así como los resultados obtenidos.

Los indicadores a comprobar son los que más inciden en la eficiencia de las búsquedas por proximidad, esto es, por una parte y de manera primordial el número de evaluaciones de la función de distancia, y por otra, el tamaño de los índices utilizados. Estos factores se comprueban tanto para las búsquedas por rango como para las búsquedas kNN.

La estrategia de Reducción de Cluster puede aplicarse a cualquier método de búsqueda por similitud basado en particiones que use el radio de cobertura como

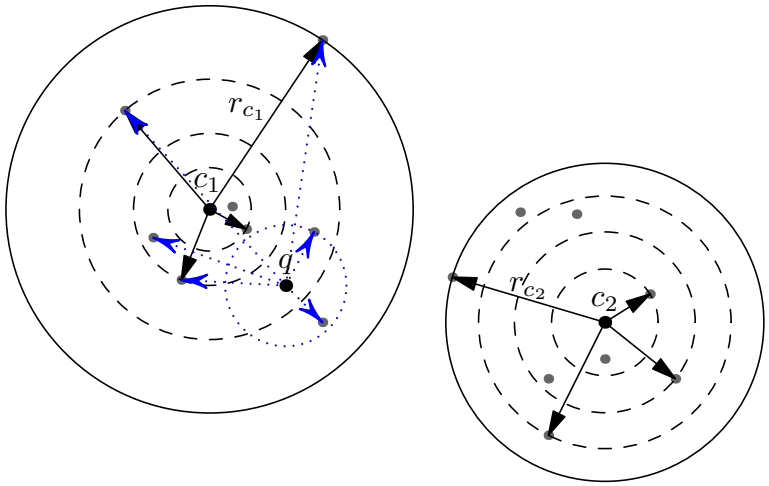


Figura 5.7: Tercer ejemplo de búsqueda kNN con Reducción de Cluster.

criterio de descarte. Las consideraciones generales del método que se utilice no varían, salvo en la definición de regiones en cada cluster y lo que eso conlleva respecto a almacenar en el índice sus características, y en la exploración de los clusters no descartados, que se realiza no sobre la totalidad del cluster, sino de forma progresiva sobre las regiones del mismo. En este trabajo se ha elegido aplicar Reducción de Cluster sobre el método de Lista de Clusters, ya que es un ejemplo representativo de la eficiencia lograda por los métodos basados en particiones que usan el radio de cobertura como criterio de descarte.

La aplicación de la estrategia de Reducción de Cluster a un método determinado, modifica tanto la estructura del índice como los algoritmos de búsqueda, que son los elementos que caracterizan a un método. Por tanto, la aplicación de la estrategia de Reducción de Cluster a un método existente, supone realmente una serie de cambios estructurales en el método.

Para contrastar el rendimiento de Reducción de Cluster se ha aplicado a las siguientes colecciones incluidas en la librería *Metric Spaces Library*[1] que es de disponibilidad pública:

- English: una colección de 69.069 palabras extraídas de un diccionario de inglés.

[1] http://sisap.org/Metric_Space_Library.html

- GERMAN: una colección de 75.086 palabras extraídas de un diccionario de alemán.
- NASA: contiene 40.150 imágenes de los archivos de la NASA, representadas por vectores de dimensión 20.
- COLORS: contiene 112.544 histogramas de colores, representados por vectores de dimensión 112.
- UNIFORM-10, UNIFORM-12: colecciones de 100.000 vectores de dimensión 10 y 12 respectivamente, distribuidos uniformemente en el cubo unidad.

Las cuatro primeras colecciones contienen datos reales y las dos restantes son sintéticas. Las dos primeras están formadas por palabras y las otras por vectores; de las de vectores, las dos primeras provienen de imágenes. Con el objetivo de ampliar las posibilidades de las pruebas se han usado dos colecciones de cada uno de los tipos: palabras, imágenes y vectores sintéticos, porque aunque se trata de objetos de la misma naturaleza, la complejidad de sus búsquedas varía.

Para que la variedad se ampliase también a la dimensionalidad, entre las colecciones de vectores se han incluido varias de dimensionalidad reducida, pero también se ha contrastado el efecto utilizando COLORS, que es un ejemplo de colección con alta dimensionalidad.

Al igual que en el capítulo anterior, el 90 % de los elementos de las colecciones se ha utilizado como objetos a indexar, mientras que el 10 % restante se ha destinado a objetos a consultar. Los objetos se han comparado utilizando la distancia de edición en las colecciones de palabras, mientras que en las de vectores se ha usado la distancia euclídea. En las búsqueda por rango se ha utilizado un rango $r = 2$ en las colecciones de palabras, mientras que en las de vectores, el que obtuviese el $0,01\,\%$ de la base de datos, en promedio para cada consulta.

En definitiva, los objetivos del análisis experimental a realizar son:

- Estudiar si la mejora obtenida en el rendimiento de las búsquedas efectuadas con Reducción de Cluster es significativa.
- Determinar la incidencia de esta estrategia en el tamaño del índice.
- Analizar las consideraciones de equilibrio que se dan entre el rendimiento de las búsquedas y el tamaño del índice.

En los siguientes apartados se presentan los resultados obtenidos en el coste de las búsquedas por rango y kNN, y un análisis del tamaño de índice frente al coste de la búsqueda.

Como veremos más adelante, las conclusiones principales del análisis son que el rendimiento obtenido al dividir los clusters en un pequeño número de regiones es un

poco menor al obtenido al dividirlo en todas las regiones posibles, y que la mejora del rendimiento de las búsquedas se sitúa entre un 13 y un 25 %, mientras que el incremento del tamaño del índice está entre un 2 y un 20 %.

5.4.1. Búsqueda por rango

Para plantear una búsqueda por rango con Reducción de Cluster, hay que determinar dos cuestiones fundamentales relacionadas con la manera en la que se realiza la división de los clusters, que son las siguientes:

- El número de regiones en las que se divide cada cluster, denotado con β.
- El número de objetos que contendrá cada cluster, al que nos referiremos como *tamaño del cluster*.

Como ya se mencionó en la sección 5.2, resulta más efectivo que las diferentes regiones tengan el mismo número de elementos. El número de regiones en las que pueden dividirse los clusters, van desde un mínimo de dos hasta un máximo determinado por el número de objetos del cluster, exceptuando el centro. Cuanto mayor sea el número de regiones, mejor rendimiento se obtendrá las búsquedas, en términos de evaluaciones de la función de distancia, pero también se necesitará un espacio mayor para almacenar el índice. Estas apreciaciones se cuantifican en esta sección.

En la experimentación se han utilizado como valores de β, o número de regiones definidas en cada cluster, los valores de 2, 5, 10, 20 y también el resultante de considerar todos los objetos menos el centro del cluster. Estos valores se han considerado más significativos para reflejar la tendencia de los resultados, después de un proceso de pruebas previo.

Respecto al número de objetos que contendrá cada cluster, o tamaño del cluster, se ha realizado también una batería de pruebas que ha establecido que los valores de 10, 20, 40, 60, 80 y 100, permiten determinar con claridad el comportamiento de la estrategia, por lo que son los utilizados en los experimentos finales.

En las figuras de la 5.8 a la 5.13 y en las tablas de la 5.1 a la 5.6, de las páginas 155 a la 160, se muestran los resultados del número medio de evaluaciones de la función de distancia, en miles, obtenidos aplicando los diferentes algoritmos a las distintas colecciones, utilizando los tamaños de cluster mencionados.

Primero se aplicó el método de Lista de Clusters original, que se denota con "`LC`". Los restantes algoritmos aplican la estrategia de Reducción de Cluster a la Lista de Clusters, diferenciándose en el número de regiones en las que dividen el cluster, y se representan mediante "`LC CR-i`" siendo $i = \beta$ el número de regiones en las que se

ha dividido el cluster, y con "`LC CR-all`" el caso de utilizar tantas regiones como objetos hay en el cluster. No se presentan valores para las combinaciones de tamaño de cluster de 10 y `LC CR-10`, ni 20 y `LC CR-20`, por coincidir ambas con `LC CR-all`, ni para el tamaño de 10 y `LC CR-20` por ser un caso imposible al tener que dividir el cluster en más zonas que el número de elementos que posee.

Las tablas asociadas a cada figura muestran los valores exactos del número de comparaciones de la función de distancia. En ellas se representa el tamaño del cluster mediante "`T.C.`"

5.4.2. Búsqueda kNN

En el caso de las búsquedas kNN con Reducción de Cluster se han utilizado los mismos planteamientos que en el caso de las búsquedas por rango, en lo referente al número de regiones en las que se dividen los clusters, considerando que β toma los valores de 2, 5, 10, 20 y también el caso en el que $\beta = all$ que se refiere a considerar todos los objetos menos el centro del cluster.

La búsqueda kNN depende también del parámetro k, que representa el número de vecinos más cercanos, por lo que las prueban se realizaron para los valores de k de 1 a 10, comprobando el efecto para los diferentes valores de β.

Aún queda por establecer el tamaño del cluster, pero dado que de esta forma se tenían ya dos dimensiones y el tamaño del cluster supondría la tercera, se consideró más adecuado fijar un tamaño de cluster determinado, que diese buenos resultados para las diferentes colecciones, y que resultó ser 40. En las pruebas previas realizadas, el comportamiento de los algoritmos fue semejante utilizando otros tamaños de cluster.

Las figuras de la 5.14 a la 5.19 y las tablas de la 5.7 a la 5.12, de las páginas 161 a la 166, reflejan el número medio de evaluaciones de la función de distancia, en miles, con respecto al número de vecinos, para las mismas colecciones y métodos que en el caso de las búsquedas por rango, utilizando las mismas etiquetas usadas allí y considerando en todos los casos un tamaño de cluster de valor 40.

Dado que cada tabla se corresponde con una figura, aparecen ambas en la misma página, con el objetivo de poder realizar un mejor seguimiento de los resultados obtenidos.

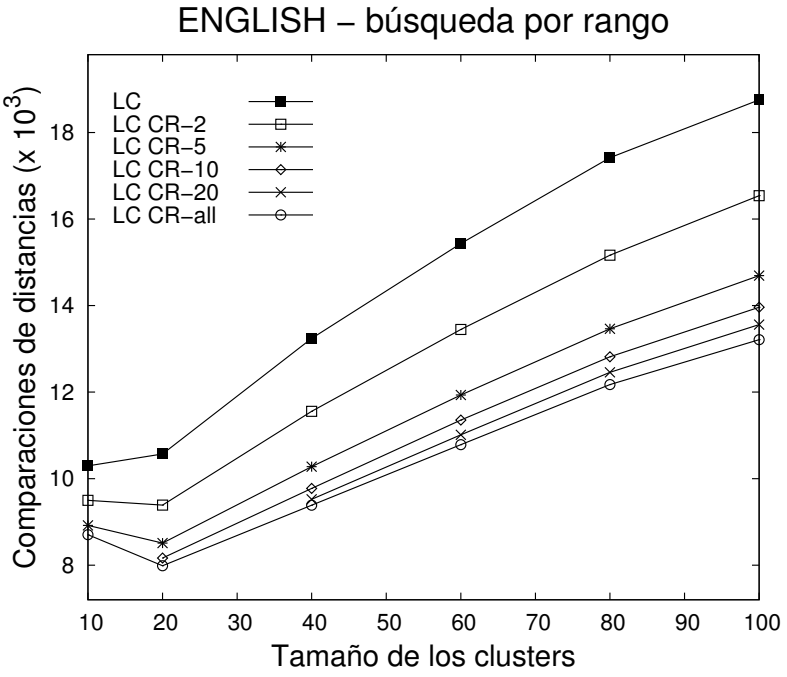


Figura 5.8: Comportamiento de la búsqueda por rango en la colección ENGLISH.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	10.294,26	9.499,06	8.919,30			8.704,01
20	10.571,08	9.388,14	8.511,02	8.165,25		7.985,73
40	13.241,74	11.555,30	10.276,90	9.774,00	9.521,68	9.385,61
60	15.433,39	13.448,86	11.933,17	11.353,08	11.014,48	10.784,20
80	17.417,09	15.163,66	13.463,18	12.817,75	12.457,11	12.173,04
100	18.755,23	16.539,96	14.692,56	13.960,21	13.559,22	13.211,04

Tabla 5.1: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre ENGLISH, según el tamaño del cluster (T. C.)

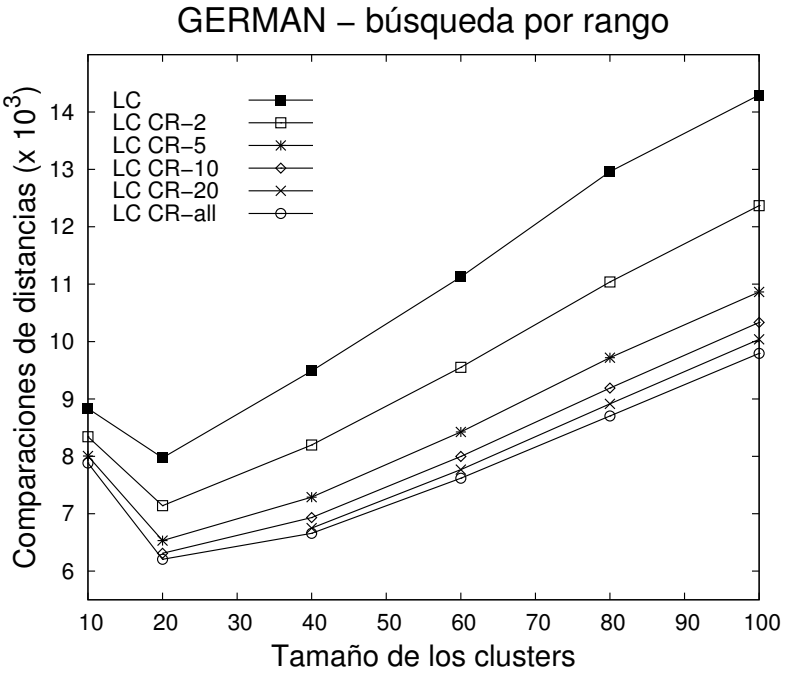


Figura 5.9: Comportamiento de la búsqueda por rango en la colección GERMAN.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	8.834,96	8.341,86	8.006,42			7.883,44
20	7.974,91	7.140,39	6.531,97	6.310,43		6.206,29
40	9.488,59	8.198,24	7.289,24	6.932,15	6.749,67	6.658,15
60	11.126,33	9.550,74	8.425,74	7.998,98	7.769,30	7.618,06
80	12.960,73	11.039,34	9.718,40	9.189,84	8.915,06	8.702,20
100	14.297,43	12.369,67	10.865,08	10.333,36	10.038,25	9.794,61

Tabla 5.2: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre GERMAN, según el tamaño del cluster (T. C.)

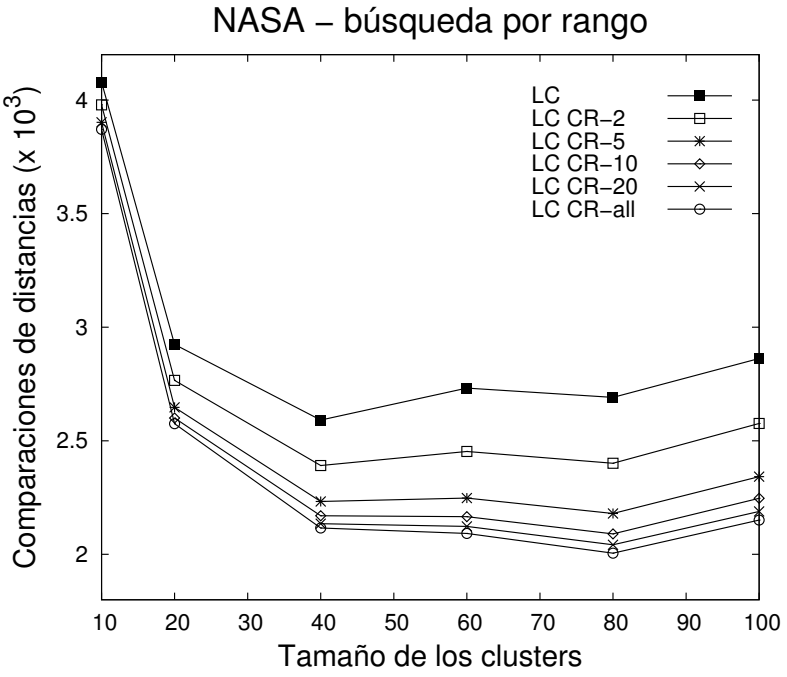


Figura 5.10: Comportamiento de la búsqueda por rango en la colección NASA.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	4.079,87	3.979,35	3.902,24			3.871,25
20	2.924,37	2.766,45	2.647,08	2.600,63		2.575,92
40	2.591,21	2.391,26	2.233,24	2.170,44	2.135,30	2.116,97
60	2.732,87	2.453,57	2.248,30	2.166,81	2.123,41	2.092,14
80	2.691,36	2.401,34	2.180,40	2.090,33	2.042,90	2.005,93
100	2.862,80	2.576,87	2.342,13	2.247,40	2.189,13	2.151,16

Tabla 5.3: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre NASA, según el tamaño del cluster (T. C.)

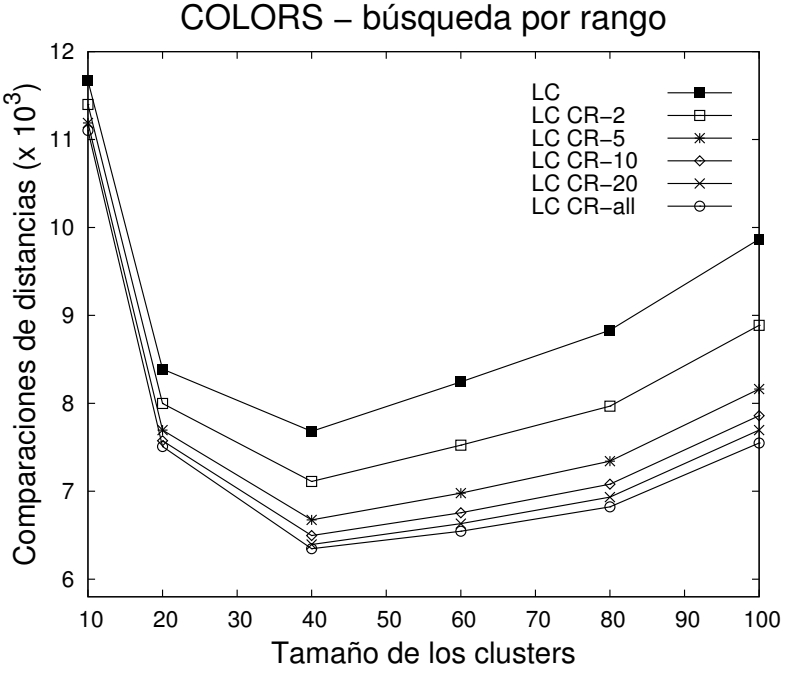


Figura 5.11: Comportamiento de la búsqueda por rango en la colección COLORS.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	11.669,73	11.399,29	11.189,94			11.104,75
20	8.389,44	7.998,90	7.696,58	7.574,58		7.508,87
40	7.682,80	7.111,70	6.674,69	6.496,93	6.395,53	6.347,41
60	8.244,20	7.524,60	6.978,80	6.756,16	6.632,13	6.545,53
80	8.832,36	7.968,42	7.342,93	7.081,63	6.933,67	6.823,16
100	9.867,33	8.887,13	8.162,24	7.860,78	7.696,39	7.548,65

Tabla 5.4: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre COLORS, según el tamaño del cluster (T. C.)

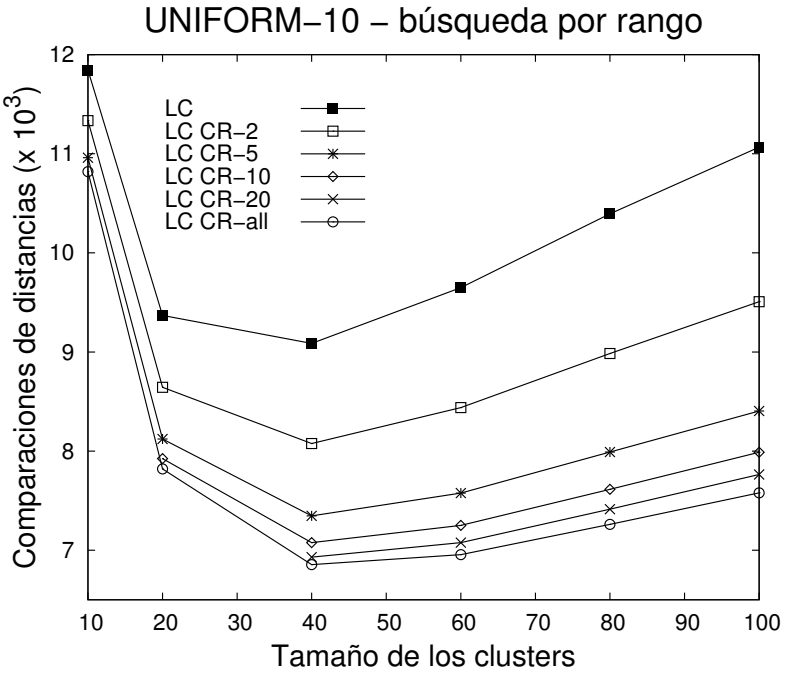


Figura 5.12: Comportamiento de la búsqueda por rango en la colección UNIFORM-10.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	11.844,32	11.334,18	10.960,11			10.819,52
20	9.369,58	8.644,27	8.122,32	7.924,33		7.820,03
40	9.086,46	8.076,14	7.347,30	7.076,94	6.930,71	6.855,96
60	9.649,79	8.439,62	7.577,27	7.250,98	7.076,40	6.955,56
80	10.394,65	8.985,95	7.991,93	7.614,06	7.414,65	7.261,99
100	11.067,24	9.507,31	8.405,95	7.988,36	7.764,72	7.579,64

Tabla 5.5: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre UNIFORM-10, según el tamaño del cluster (T. C.)

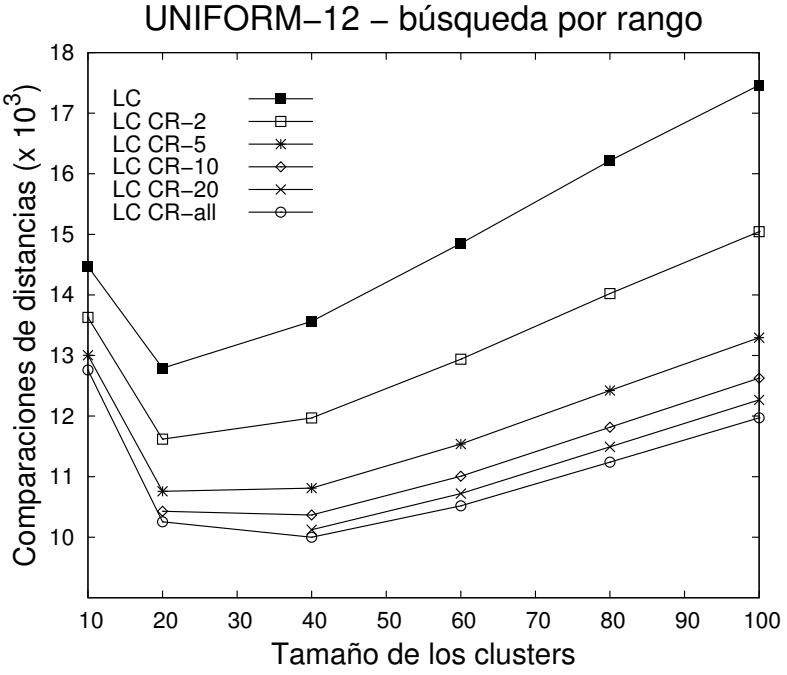


Figura 5.13: Comportamiento de la búsqueda por rango en la colección UNIFORM-12.

T. C.	LC	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
10	14.470,77	13.630,08	13.004,29			12.761,72
20	12.790,67	11.618,36	10.758,42	10.429,89		10.254,12
40	13.564,39	11.968,78	10.810,14	10.366,03	10.124,39	10.000,51
60	14.850,86	12.939,05	11.539,65	11.006,93	10.720,18	10.519,51
80	16.216,13	14.022,21	12.421,76	11.816,86	11.492,70	11.238,50
100	17.462,23	15.045,01	13.295,32	12.625,15	12.266,76	11.970,98

Tabla 5.6: Número de evaluaciones de la función de distancia obtenido en la búsqueda por rango sobre UNIFORM-12, según el tamaño del cluster (T. C.)

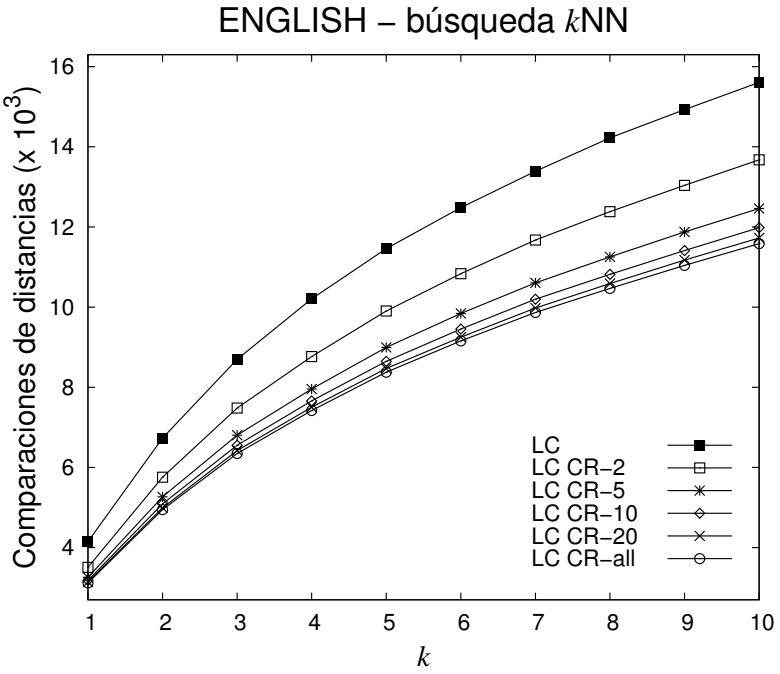


Figura 5.14: Comportamiento de la búsqueda kNN en la colección ENGLISH.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	4.150,04	3.509,02	3.259,82	3.185,20	3.144,48	3.121,13
2	6.733,74	5.757,20	5.268,54	5.092,63	4.994,60	4.949,12
3	8.699,34	7.482,22	6.806,25	6.554,90	6.423,00	6.350,98
4	10.207,33	8.767,92	7.957,26	7.660,30	7.502,37	7.420,12
5	11.459,35	9.905,58	8.996,67	8.650,45	8.469,08	8.377,06
6	12.483,30	10.834,20	9.842,53	9.454,39	9.251,49	9.157,43
7	13.390,80	11.675,04	10.607,35	10.199,71	9.978,78	9.869,20
8	14.224,31	12.381,13	11.254,06	10.817,36	10.590,24	10.467,61
9	14.929,10	13.038,39	11.876,91	11.413,42	11.174,03	11.041,40
10	15.606,45	13.676,25	12.461,86	11.983,61	11.726,73	11.585,48

Tabla 5.7: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre ENGLISH.

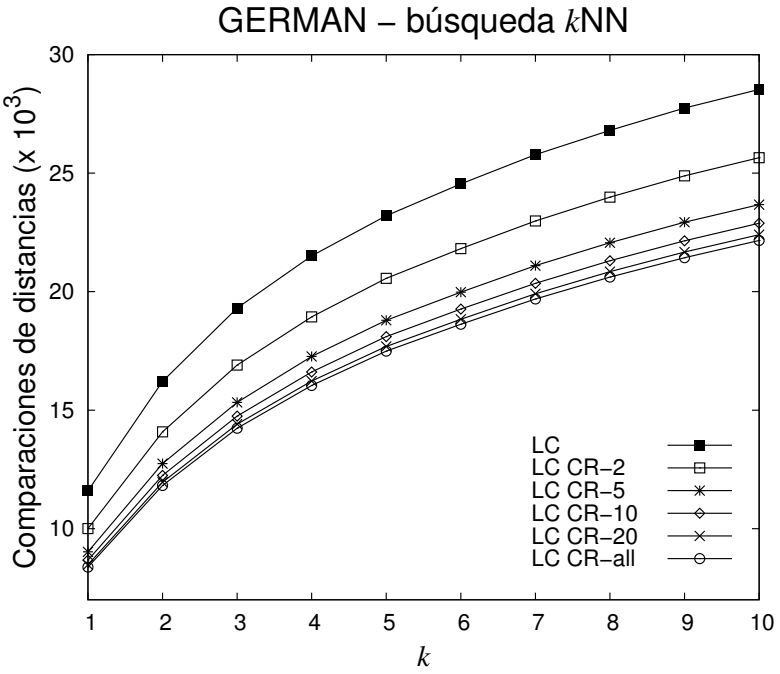


Figura 5.15: Comportamiento de la búsqueda kNN en la colección GERMAN.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	11.597,40	10.006,20	9.025,42	8.676,87	8.478,46	8.380,14
2	16.215,89	14.086,47	12.743,10	12.247,72	11.959,67	11.821,60
3	19.310,80	16.903,87	15.330,87	14.751,21	14.411,10	14.238,16
4	21.508,20	18.934,17	17.264,16	16.612,62	16.229,93	16.042,59
5	23.200,02	20.561,10	18.794,80	18.103,55	17.689,91	17.492,63
6	24.545,68	21.815,72	19.980,12	19.264,63	18.834,61	18.624,44
7	25.783,10	22.981,23	21.096,71	20.352,79	19.909,52	19.691,50
8	26.806,95	23.987,04	22.062,38	21.301,76	20.842,82	20.613,21
9	27.746,20	24.886,22	22.929,80	22.143,26	21.670,79	21.433,20
10	28.530,10	25.651,57	23.673,16	22.888,76	22.403,84	22.161,42

Tabla 5.8: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre GERMAN.

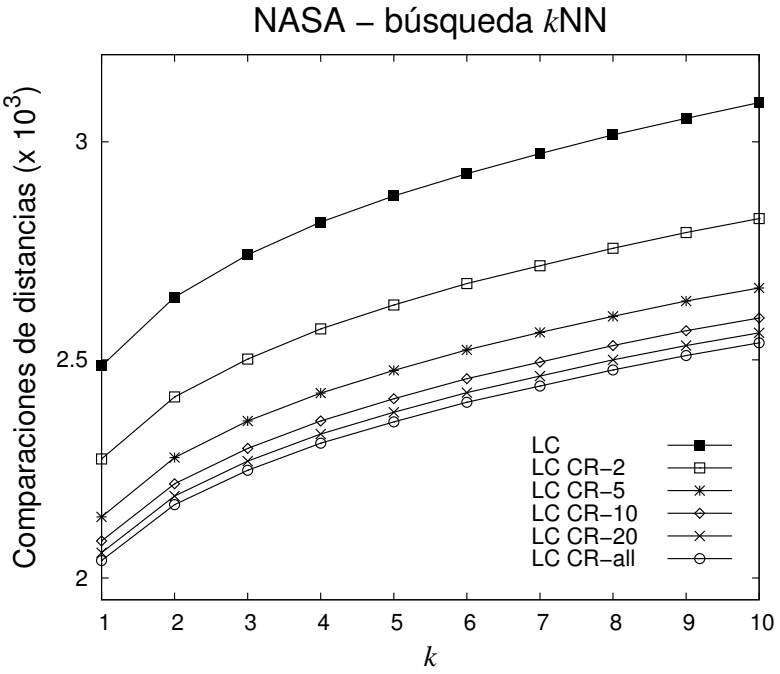


Figura 5.16: Comportamiento de la búsqueda kNN en la colección NASA.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	2.486,94	2.273,91	2.140,36	2.085,35	2.058,08	2.040,29
2	2.644,66	2.415,83	2.276,59	2.216,76	2.188,06	2.168,47
3	2.741,62	2.502,75	2.360,14	2.297,82	2.268,05	2.247,61
4	2.816,90	2.571,24	2.424,95	2.360,95	2.330,39	2.309,13
5	2.876,14	2.626,72	2.476,94	2.411,74	2.380,48	2.358,75
6	2.927,92	2.675,52	2.523,76	2.457,38	2.425,35	2.403,21
7	2.973,17	2.716,82	2.563,18	2.495,90	2.463,46	2.440,95
8	3.016,22	2.756,48	2.600,98	2.533,15	2.500,10	2.477,14
9	3.054,46	2.792,76	2.635,19	2.567,02	2.533,63	2.510,30
10	3.090,34	2.824,67	2.665,56	2.596,77	2.562,66	2.539,11

Tabla 5.9: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre NASA.

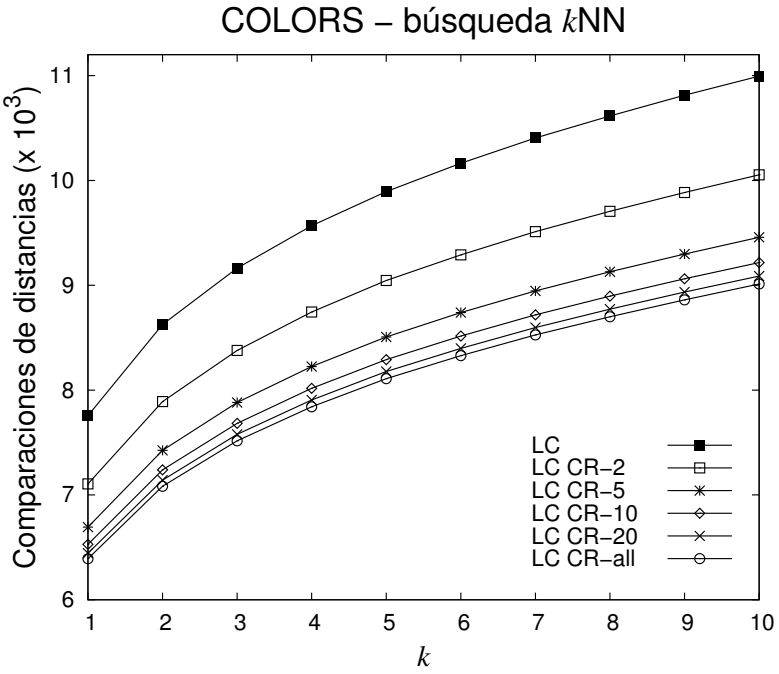


Figura 5.17: Comportamiento de la búsqueda kNN en la colección COLORS.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	7.756,27	7.105,07	6.693,36	6.529,91	6.444,84	6.392,90
2	8.626,17	7.890,08	7.425,02	7.239,36	7.141,66	7.083,05
3	9.163,32	8.378,04	7.882,60	7.683,73	7.578,64	7.516,08
4	9.568,69	8.745,04	8.225,31	8.017,02	7.906,42	7.841,08
5	9.894,96	9.046,33	8.507,90	8.291,92	8.177,05	8.109,28
6	10.164,95	9.290,72	8.739,44	8.516,93	8.398,41	8.328,80
7	10.405,55	9.512,14	8.946,80	8.718,93	8.597,36	8.526,11
8	10.615,94	9.705,72	9.129,21	8.896,70	8.772,41	8.699,83
9	10.813,53	9.885,96	9.298,97	9.062,04	8.935,13	8.861,13
10	10.994,78	10.054,46	9.458,18	9.217,29	9.088,01	9.012,72

Tabla 5.10: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre COLORS.

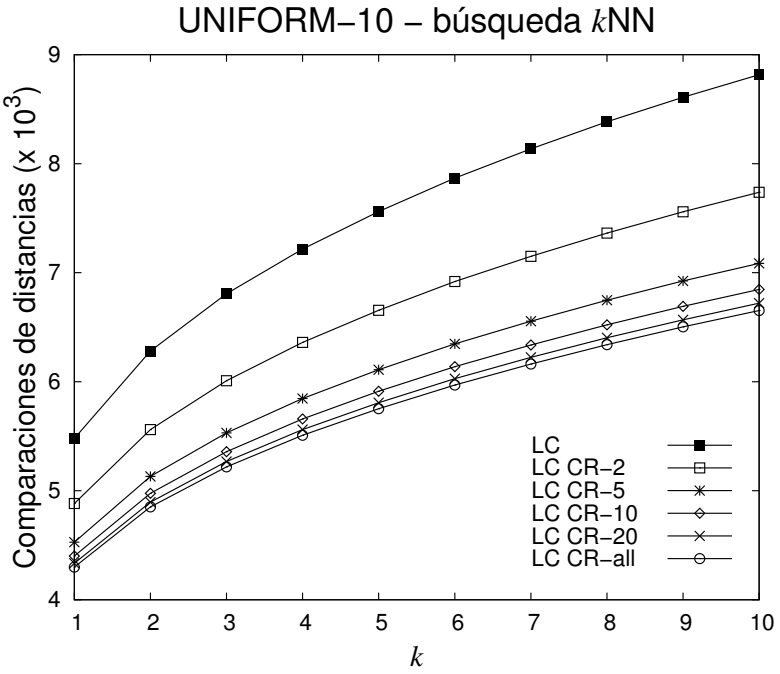


Figura 5.18: Comportamiento de la búsqueda kNN en la colección UNIFORM-10.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	5.480,26	4.883,03	4.528,57	4.401,63	4.335,23	4.299,76
2	6.282,21	5.561,97	5.131,78	4.976,71	4.895,45	4.852,14
3	6.807,94	6.009,33	5.531,96	5.358,99	5.268,02	5.219,66
4	7.217,02	6.361,21	5.846,72	5.659,73	5.561,47	5.509,30
5	7.561,07	6.655,90	6.111,00	5.912,56	5.807,94	5.752,50
6	7.868,62	6.919,33	6.347,77	6.138,90	6.028,98	5.970,72
7	8.136,21	7.150,25	6.555,33	6.337,87	6.223,10	6.162,30
8	8.384,00	7.363,70	6.747,76	6.522,31	6.403,15	6.340,03
9	8.609,81	7.559,93	6.924,26	6.691,47	6.568,53	6.503,29
10	8.816,26	7.738,84	7.086,52	6.846,77	6.720,37	6.653,27

Tabla 5.11: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre UNIFORM-10.

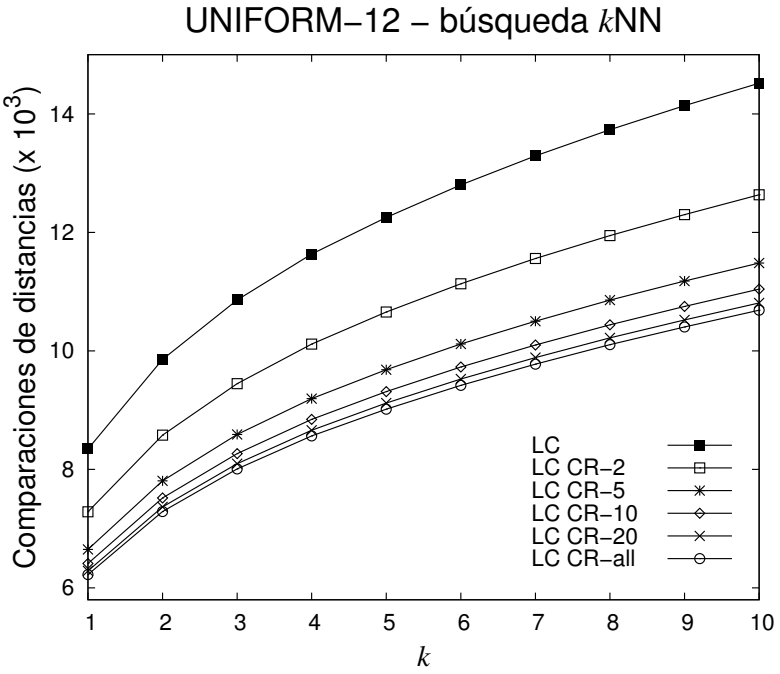


Figura 5.19: Comportamiento de la búsqueda kNN en la colección UNIFORM-12.

k	**LC**	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
1	8.344,07	7.285,77	6.650,36	6.412,98	6.288,88	6.224,52
2	9.856,45	8.579,30	7.808,77	7.519,19	7.367,36	7.288,17
3	10.860,92	9.447,49	8.590,63	8.266,93	8.096,97	8.008,31
4	11.634,41	10.115,94	9.194,43	8.845,40	8.661,78	8.565,75
5	12.252,55	10.656,09	9.682,46	9.313,76	9.119,09	9.017,43
6	12.802,95	11.133,49	10.116,15	9.729,36	9.525,35	9.418,52
7	13.291,54	11.559,07	10.502,86	10.100,04	9.887,68	9.776,29
8	13.734,74	11.947,56	10.857,36	10.440,49	10.220,42	10.105,00
9	14.139,77	12.301,50	11.179,54	10.750,39	10.523,52	10.404,58
10	14.518,65	12.634,43	11.483,79	11.042,88	10.809,59	10.687,26

Tabla 5.12: Número de evaluaciones de la función de distancia obtenido en la búsqueda kNN sobre UNIFORM-12.

5.4.3. Requerimientos de espacio

El rendimiento de las búsquedas por similitud viene determinado por el número de evaluaciones de la función de distancia, pero además es necesario considerar el tamaño del índice para comprobar la eficiencia de los métodos.

La tabla 5.13 muestra el tamaño del índice al usar el método Lista de Clusters original y al aplicarle Reducción de Cluster, con un tamaño de cluster de 40, para los diferentes números de regiones utilizadas en la experimentación con las colecciones reales. La columna "Relativo" indica el tamaño relativo del índice, también con respecto a Lista de Clusters original. Se aprecia un incremento del tamaño del índice de un 2 % al considerar dos regiones, de un 9 % al definir 5, y de un 20 % si se consideran 10 regiones. A partir de este número, el tamaño del índice sufre un crecimiento considerable, llegando al 89 % si se consideran todos los objetos del cluster. Estos porcentajes de crecimiento del índice son idénticos en todas las colecciones, por lo que solo aparecen una vez en la tabla.

La figura 5.20 refleja la relación entre la mejora del coste de la búsqueda por rango y el incremento del espacio ocupado por el índice, para un tamaño de cluster igual a 40, expresando los resultados en valores relativos de porcentajes comparados con el método original de Lista de Clusters. Los valores del eje de abscisas se corresponden con los de la última columna de la tabla 5.13. Se muestran todas las colecciones utilizadas y el efecto de realizar las divisiones de los clusters en los números de regiones considerados, esto es, para los valores de β iguales a 2, 5, 10, 20 y también el obtenido al usar todos los objetos de los clusters. Por ejemplo, para la colección German al usar un $\beta = 2$, esto es, dos regiones, se genera un incremento del espacio del índice de un 2 % (eje OX), como se refleja en la tabla 5.13, y se produce una mejora del coste de la búsqueda del 13,59 % (eje OY), obtenida en función de la reducción del número de evaluaciones de la función de distancia, ya que se pasa de 9.488,59 a 8.198,24, valores que se reflejan en las columnas "`LC`" y "`LC CR-2`" de la tabla 5.2, para el tamaño de cluster igual a 40.

	English	**German**	**Nasa**	**Colors**	**Relativo**
LC	260,65	283,34	151,53	424,66	1,00
$\beta = 2$	266,57	289,79	154,97	434,31	1,02
$\beta = 5$	284,34	309,10	165,29	463,26	1,09
$\beta = 10$	313,95	341,29	182,50	511,50	1,20
$\beta = 20$	373,17	405,66	216,92	607,99	1,43
$\beta = all$	491,60	534,41	285,74	800,96	1,89

Tabla 5.13: Requerimientos de espacio del índice (en KB) de Lista de Clusters (LC) y al aplicarle Reducción de Cluster según el número de regiones en las que se dividen los clusters (β) y considerando un tamaño de cluster de 40.

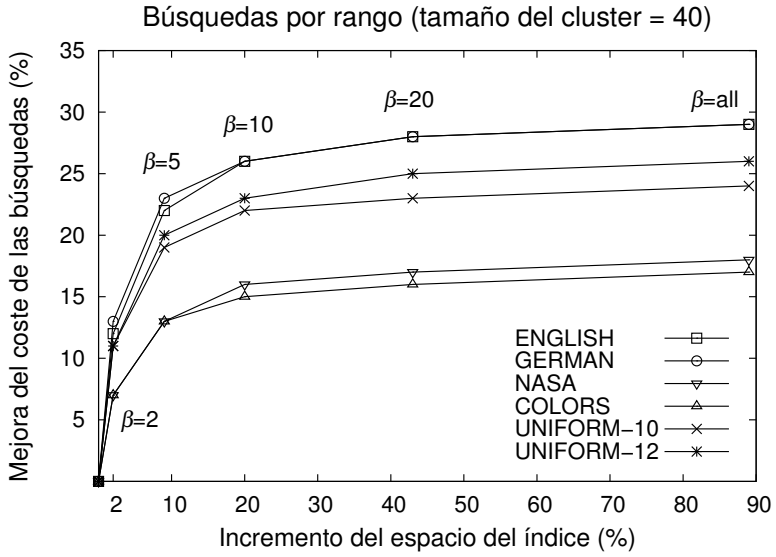


Figura 5.20: Equilibrio entre la mejora del coste de la búsqueda por rango frente al incremento del espacio del índice, considerando el tamaño del cluster igual a 40 y dividiendo los clusters en los siguientes números de regiones: 2, 5, 10, 20 y todas las posibles regiones (*all*).

5.5. Discusión

En esta sección se analizan los resultados obtenidos en la evaluación experimental de los algoritmos de Reducción de Cluster.

5.5.1. Búsqueda por rango

En lo referente a las búsquedas por rango, los resultados obtenidos muestran el importante impacto que se produce en el rendimiento de las búsquedas aplicando Reducción de Cluster, ya que el número de evaluaciones de la función de distancia se reduce considerablemente al realizar la exploración de los clusters en función de sus regiones y no de la totalidad de los mismos, reduciendo así la complejidad externa.

La mejora se produce para todas las colecciones, en todos los tamaños de cluster evaluados y para los diversos números de regiones en las que se han dividido los clusters. El impacto varía dependiendo de la colección y del tamaño de los clusters, ya que en unas colecciones los valores mínimos de las evaluaciones de la función de distancia se obtienen para un tamaño de cluster y en otras para otro, pero en

general, la mejora del rendimiento se produce de manera similar para las diferentes colecciones y en todos los tamaños de clusters.

Lo primero que se deduce del análisis de las figuras de la 5.8 a la 5.13 y en las tablas de la 5.1 a la 5.6, de las páginas 155 a la 160, es que el tamaño de los clusters tiene un efecto importante en el número de evaluaciones de la función de distancia obtenidas. Esto ocurre para Lista de Clusters y se reproduce al aplicar Reducción de Cluster para los distintos números de regiones en las que se dividen los clusters, esto es, para los diferentes valores de β, por lo que las gráficas de Reducción de Cluster tienen una forma similar que las de Lista de Clusters. Esta circunstancia se da en todas las colecciones analizadas.

Los valores mínimos del número de comparaciones de la función de distancia se dan para los tamaños de cluster de 20, en el caso de las colecciones English, German y Uniform-12, y para las restantes en 40, salvo para la colección Nasa. La aplicación de Reducción de Cluster mantiene ese comportamiento en todos los casos y para todos los valores de β, excepto en Uniform-12, que desplaza el mínimo al tamaño de cluster 40 a partir del valor de $\beta = 10$.

Se observa que utilizando valores intermedios del tamaño del cluster, entre 20 y 40, se obtienen los valores mínimos de la evaluación de las funciones de distancia, salvo en Nasa. La estrategia Reducción de Cluster mantiene en general ese comportamiento. Para valores del tamaño del cluster menores que 20 o mayores de 80, el número de evaluaciones tiende a crecer.

La aplicación de Reducción de Cluster logra una disminución significativa del número de evaluaciones de la función de distancia, dependiendo en gran medida del número de regiones en las que se han dividido los clusters y también del tamaño de esos clusters. Con solo dos regiones ya se obtienen mejoras importantes, que sobrepasan el 10 % en la mayoría de los casos, y que llegan a estar entre el 12 y el 14 % en buena parte de los resultados. La tabla 5.14 de la página 173, obtenida a partir de los datos de las tablas 5.1 a la 5.6, refleja los porcentajes de mejora del número de evaluaciones de la función de distancia, al aplicar la estrategia de Reducción de Cluster al método Lista de Clusters. Por ejemplo, en la colección German con un tamaño de cluster de 40 y dividiendo los clusters en cinco regiones (`LCCR=5`), la mejora de aplicar Reducción de Cluster es del 23,18 % con respecto a LC. La tabla permite observar la evolución de los porcentajes a medida que varía, tanto el número de regiones, como el tamaño del cluster.

Al incrementar el número de regiones, el rendimiento de las búsquedas mejora, llegando a ser máximo al utilizar todos los objetos de los cluster ($\beta = all$). De esta forma se obtienen porcentajes de mejora del rendimiento que superan en muchos casos el 10 % con solo dos regiones, que están en un rango de 10 a 25 % para cinco regiones, que casi siempre sobrepasan el 25 % con diez regiones, y que en algunos casos llegan a alcanzar el 30 %, considerando todos los objetos del cluster.

El impacto de la estrategia propuesta queda patente incluso al dividir los clusters en un pequeño número de regiones. Se observa que con solo dos regiones se obtienen porcentajes que son aproximadamente la mitad de los mejores que se pueden lograr. También es destacable el hecho de que usando cinco regiones los resultados son ya bastante cercanos a los mejores posibles. En el caso de usar diez regiones, el número de evaluaciones de la función de distancia está ya muy cerca del mejor valor que se puede obtener. A partir de 10 o 20 regiones, incrementar su número todavía mejora el rendimiento, pero lo hace de una manera muy reducida. Estas circunstancias se observan también en todas las colecciones, resultando trascendental cuando posteriormente se analicen las consideraciones de equilibrio entre el rendimiento de las búsquedas y el tamaño del índice.

5.5.2. Búsqueda kNN

En la búsqueda kNN se realizaron las pruebas considerando un tamaño de cluster fijo de valor 40, con lo que los resultados quedan en función del número de regiones en las que se dividen los clusters y del número de k vecinos más cercanos a buscar.

El número de vecinos a buscar condiciona los resultados de manera importante, ya que el número de evaluaciones de la función de distancia crece progresivamente a medida que se incrementa el valor de k, puesto que hay que realizar más búsquedas para encontrar más vecinos.

Los resultados muestran un comportamiento similar al obtenido en las búsquedas por rango. Por una parte, el rendimiento mejora a medida que se utiliza un número mayor de regiones por cluster, siendo máximo cuando intervienen todos los objetos del mismo, haciendo que cada uno genere una división de las regiones del cluster. Al igual que ocurría en las búsquedas por rango, el impacto de la Reducción de Cluster en las búsquedas kNN es notorio usando incluso un número muy reducido de regiones, algo que se aprecia con claridad en las figuras de la 5.14 a la 5.19 y en las tablas de la 5.7 a la 5.12, de las páginas 161 a la 166.

En particular se observa que la gráfica correspondiente a la división en dos regiones, denotada con "`LC CR-2`", presenta valores bastante más bajos que la del método Lista de Clusters original, etiquetada con "`LC`". Este hecho se reproduce de manera análoga en todas las colecciones y para todos los valores de k; incluso en las colecciones de palabras, en las que para el valor de $k = 1$ las gráficas indican una pequeña variación entre "`LC`" y "`LC CR-all`", resulta que la división de los clusters en dos regiones obtiene los mejores porcentajes de incremento del rendimiento para las distintas colecciones en el caso de considerar dos regiones. Este hecho se comprueba al observar los valores porcentuales de incremento del rendimiento de las búsquedas kNN, que se muestran en las tablas 5.15 y 5.16 de las páginas 174 y 175, respectivamente.

La búsqueda kNN presenta incluso un comportamiento más uniforme que la búsqueda por rango, ya que los porcentajes de mejora del rendimiento de las búsquedas son más parecidos, tanto al considerar la evolución del valor de k, como el del número de regiones. Esta circunstancia explica también la mayor regularidad que presentan las gráficas en este tipo de búsqueda.

Considerando cinco regiones se obtienen resultados muy significativos, y a partir de diez, son muy cercanos a los mejores posibles. A partir de ahí, incrementar el número de regiones produce mejoras muy poco significativas en el rendimiento de las búsquedas. Los resultados son homogéneos para todos los valores de k, en todas las colecciones y para los diferentes números de regiones consideradas.

Tanto de estos resultados como de los obtenidos en las búsquedas por rango, se puede afirmar que la estrategia de Reducción de Cluster produce una mejora significativa del rendimientos de las búsquedas por similitud, ya que logra una importante reducción del número de evaluaciones de la función de distancia al explorar los clusters no descartados, lo que supone una reducción de la complejidad externa. Este hecho se produce incluso al considerar una división de los clusters en un número muy reducido de regiones, por ejemplo solo dos, dándose además la circunstancia de que la mejora del rendimiento crece de manera más rápida al considerar un número pequeño de regiones, como cinco o diez, para a partir de ahí seguir tendiendo al valor máximo pero con un crecimiento muy tenue.

5.5.3. Rendimiento de las búsquedas y requerimientos de espacio

La estrategia propuesta produce una mejora del rendimiento de las búsquedas, pero para su utilización efectiva se necesita comprobar con rigurosidad el espacio que requieren sus estructuras auxiliares.

La tabla 5.13 de la página 167 indica el tamaño de los índices de las colecciones reales, para un tamaño de cluster de 40. Refleja también el incremento del espacio que origina la estrategia de Reducción de Cluster, con respecto a Lista de Clusters original, en función de los diferentes números de regiones consideradas, que aparece en la columna “Relativo”. El análisis de estos datos es fundamental para realizar una evaluación completa de la estrategia propuesta.

Se observa que existe una relación directa entre el número de regiones en las que se dividen los clusters y el tamaño del índice, ya que a medida que se incrementa el número de regiones, aumenta el espacio ocupado por el índice. Este crecimiento es similar en las colecciones estudiadas, hasta el punto de que es igual en términos porcentuales en todas ellas.

Se tiene por ejemplo, que al considerar dos regiones, el índice crece un 2 % en todas las colecciones reales, lo que supone un valor muy reducido. Si se dividen los clusters en cinco regiones, el índice crece un 9 %, y si se consideran diez regiones, el índice incrementa su tamaño en un 20 %. Estos valores están entre los que normalmente resultan asumibles para las estructuras auxiliares, si bien en el caso de diez regiones alcanza ya un valor que suele considerarse como límite. Se ha estudiado el efecto que provoca la división en otros números de regiones, pero en ellos el tamaño del índice alcanza ya unos valores inaceptables; es el caso de considerar veinte regiones, con un incremento de tamaño del 43 %, y también la totalidad de los objetos del cluster, que requiere un espacio adicional de un 89 %.

De todo lo anterior se deduce que la estrategia de Reducción de Cluster podría ser aplicable, en términos únicamente de espacio ocupado por el índice, si el número de regiones en las que se dividen los clusters es menor o igual a diez, y que cuanto menor sea ese valor, resultará más adecuada ya que el tamaño del índice será menor.

Por otra parte, en el análisis de las búsquedas se ha comprobado que existe una relación directa entre el rendimiento de las consultas y el número de regiones en las que se dividen los clusters, ya que el primero se incrementa a medida que aumenta el segundo. Combinando todos los factores, se tiene que si el rendimiento de las consultas se incrementa, también lo hace el tamaño del índice, por lo que parece que hay un equilibrio entre el espacio adicional necesario en el índice y la reducción del número de evaluaciones de la función de distancia.

El hecho que da validez a la estrategia viene dado por tener en cuenta conjuntamente los porcentajes de crecimiento del índice con los del incremento del rendimiento de las consultas, a medida que aumenta el número de regiones en las que se dividen los clusters, ya que con un número reducido de regiones, entre cinco y diez, se obtienen mejoras importantes del rendimiento de las búsquedas, a la vez que el tamaño del índice se incrementa en porcentajes mucho menores. Esto se ha avanzado ya al final de las secciones 5.5.1 y 5.5.2, en el sentido de que al dividir los clusters en un número reducido de regiones, el impacto sobre el rendimiento de las búsquedas es considerable. Se comprobó empíricamente que usando cinco regiones se obtiene una mejora del rendimiento muy significativa y cercana ya al mejor valor posible, mientras que en el caso de utilizar diez regiones el número de evaluaciones de la función de distancia se encuentra muy próximo al que se obtendría si se considerasen todos los objetos de los clusters. La figura 5.20 de la página 168, referente a las búsquedas por rango y ya analizada al final de la sección 5.5.3, muestra claramente la situación para un tamaño de cluster de 40.

De todo lo anterior se deduce que la estrategia de Reducción de Cluster es aplicable de manera efectiva a las búsquedas por similitud sobre las colecciones estudiadas, si el número de regiones en las que se dividen los clusters es un valor reducido, por ejemplo entre cinco y diez.

Colección	T.C.	LCCR-2	LCCR-5	LCCR-10	LCCR-20	LCCR-all
ENGLISH	10	7,72	13,36			15,45
	20	11,19	19,49	22,76		24,46
	40	12,74	22,39	26,19	28,09	29,12
	60	12,86	22,68	26,44	28,63	30,12
	80	12,94	22,70	26,41	28,48	30,11
	100	11,81	21,66	25,57	27,70	29,56
GERMAN	10	5,58	9,38			10,77
	20	10,46	18,09	20,87		22,18
	40	13,60	23,18	26,94	28,87	29,83
	60	14,16	24,27	28,11	30,17	31,53
	80	14,82	25,02	29,09	31,21	32,86
	100	13,48	24,01	27,73	29,79	31,49
NASA	10	2,46	4,35			5,11
	20	5,40	9,48	11,07		11,92
	40	7,72	13,81	16,24	17,59	18,30
	60	10,22	17,73	20,71	22,30	23,45
	80	10,78	18,99	22,33	24,09	25,47
	100	9,99	18,19	21,50	23,53	24,86
COLORS	10	2,32	4,11			4,84
	20	4,66	8,26	9,71		10,50
	40	7,43	13,12	15,44	16,76	17,38
	60	8,73	15,35	18,05	19,55	20,60
	80	9,78	16,86	19,82	21,50	22,75
	100	9,93	17,28	20,34	22,00	23,50
UNIFORM-10	10	4,31	7,47			8,65
	20	7,74	13,31	15,42		16,54
	40	11,12	19,14	22,12	23,72	24,55
	60	12,54	21,48	24,86	26,67	27,92
	80	13,55	23,11	26,75	28,67	30,14
	100	14,10	24,05	27,82	29,84	31,51
UNIFORM-12	10	5,81	10,13			11,81
	20	9,17	15,89	18,46		19,83
	40	11,76	20,31	23,58	25,36	26,27
	60	12,87	22,30	25,88	27,81	29,17
	80	13,53	23,40	27,13	29,13	30,70
	100	13,84	23,86	27,70	29,75	31,45

Tabla 5.14: Porcentaje de mejora del rendimiento en las búsquedas por rango al aplicar Reducción de Cluster al método Lista de Clusters, según el tamaño de cluster (T.C.)

Colección	k	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
ENGLISH	1	15,45	21,45	23,25	24,23	24,79
	2	14,50	21,76	24,37	25,83	26,50
	3	13,99	21,76	24,65	26,17	26,99
	4	14,10	22,04	24,95	26,50	27,31
	5	13,56	21,49	24,51	26,09	26,90
	6	13,21	21,15	24,26	25,89	26,64
	7	12,81	20,79	23,83	25,48	26,30
	8	12,96	20,88	23,95	25,55	26,41
	9	12,66	20,44	23,55	25,15	26,04
	10	12,37	20,15	23,21	24,86	25,76
GERMAN	1	13,72	22,18	25,18	26,89	27,74
	2	13,13	21,42	24,47	26,25	27,10
	3	12,46	20,61	23,61	25,37	26,27
	4	11,97	19,73	22,76	24,54	25,41
	5	11,37	18,99	21,97	23,75	24,60
	6	11,12	18,60	21,52	23,27	24,12
	7	10,87	18,18	21,06	22,78	23,63
	8	10,52	17,70	20,54	22,25	23,10
	9	10,31	17,36	20,19	21,90	22,75
	10	10,09	17,02	19,77	21,47	22,32
NASA	1	8,57	13,94	16,15	17,24	17,96
	2	8,65	13,92	16,18	17,27	18,01
	3	8,71	13,91	16,19	17,27	18,02
	4	8,72	13,91	16,19	17,27	18,03
	5	8,67	13,88	16,15	17,23	17,99
	6	8,62	13,80	16,07	17,16	17,92
	7	8,62	13,79	16,05	17,14	17,90
	8	8,61	13,77	16,02	17,11	17,87
	9	8,57	13,73	15,96	17,05	17,82
	10	8,60	13,75	15,97	17,08	17,84

Tabla 5.15: Porcentaje de mejora del rendimiento en las búsquedas kNN al aplicar Reducción de Cluster al método Lista de Clusters, para un tamaño de cluster de 40 (primera parte).

Colección	k	**LCCR-2**	**LCCR-5**	**LCCR-10**	**LCCR-20**	**LCCR-all**
COLORS	1	8,40	13,70	15,81	16,91	17,58
	2	8,53	13,92	16,08	17,21	17,89
	3	8,57	13,98	16,15	17,29	17,98
	4	8,61	14,04	16,22	17,37	18,05
	5	8,58	14,02	16,20	17,36	18,05
	6	8,60	14,02	16,21	17,38	18,06
	7	8,59	14,02	16,21	17,38	18,06
	8	8,57	14,00	16,19	17,37	18,05
	9	8,58	14,01	16,20	17,37	18,06
	10	8,55	13,98	16,17	17,34	18,03
UNIFORM-10	1	10,90	17,37	19,68	20,89	21,54
	2	11,46	18,31	20,78	22,07	22,76
	3	11,73	18,74	21,28	22,62	23,33
	4	11,86	18,99	21,58	22,94	23,66
	5	11,97	19,18	21,80	23,19	23,92
	6	12,06	19,33	21,98	23,38	24,12
	7	12,12	19,43	22,10	23,51	24,26
	8	12,17	19,52	22,21	23,63	24,38
	9	12,19	19,58	22,28	23,71	24,47
	10	12,22	19,62	22,34	23,77	24,53
UNIFORM-12	1	12,68	20,30	23,14	24,63	25,40
	2	12,96	20,78	23,71	25,25	26,06
	3	13,01	20,90	23,88	25,45	26,26
	4	13,05	20,97	23,97	25,55	26,38
	5	13,03	20,98	23,99	25,57	26,40
	6	13,04	20,99	24,01	25,60	26,43
	7	13,03	20,98	24,01	25,61	26,45
	8	13,01	20,95	23,98	25,59	26,43
	9	13,00	20,94	23,97	25,57	26,42
	10	12,98	20,90	23,94	25,55	26,39

Tabla 5.16: Porcentaje de mejora del rendimiento en las búsquedas kNN al aplicar Reducción de Cluster al método Lista de Clusters, para un tamaño de cluster de 40 (segunda parte).

5.6. Características dinámicas

Un método de búsqueda por similitud tiene características dinámicas si posee capacidad para realizar operaciones de inserción, borrado y actualización de los objetos de las colecciones. Estas características son indispensables para que sea aplicable a problemas y situaciones reales.

A continuación se describen las operaciones de borrado y de inserción utilizando Reducción de Cluster, ya que la actualización puede considerarse como un borrado seguido de una inserción.

La estrategia de Reducción de Cluster se ha implementado intentando que la estructura del índice ocupase el menor espacio posible. Este condicionante ha determinado la elección de una estructura compuesta, para cada cluster, por dos arrays, uno para almacenar las distancias del centro a los objetos que determinan las regiones del cluster, y otro que contiene los identificadores enteros de los objetos del cluster. El array de enteros utiliza una correspondencia posicional acorde al número de objetos en cada región. Su primer identificador se corresponde con el objeto cuya distancia al centro del cluster determina el radio de cobertura, los siguientes son los correspondientes a los objetos que pertenecen a la región más alejada del centro, hasta que se llega al identificador asociado al objeto que determina la segunda región más alejada del centro, y así sucesivamente.

En estas condiciones la operación de borrado puede realizarse diferenciando las siguientes situaciones, según las características del objeto a borrar:

- **Un objeto diferente al centro y que no delimita ninguna región.**

  La información existente en el índice sobre dicho objeto es solo una referencia de a qué región pertenece, que se deduce de su posición en el array de enteros, por lo que su borrado consiste en encontrar esa referencia y anularla mediante un borrado lógico.

- **Un objeto que delimita una región.**

  Al tratarse de un objeto que delimita una región, al eliminar el objeto se perdería también su distancia al centro y con ella los límites de la región. Por ello se realiza un borrado lógico sobre el objeto, de forma que no aparezca como resultado de ninguna búsqueda, pero que a la vez se mantenga su distancia al centro del cluster para determinar los límites de la región.

  Un caso particular se produce si el objeto que se desea borrar es el que tiene como distancia al centro del cluster el radio de cobertura, siendo por tanto el que determina la región más exterior del cluster. Sin embargo, el borrado en este caso es similar al de los objetos que delimitan las otras regiones.

- **El centro del cluster.**

  El centro del cluster juega un papel fundamental ya que todas las regiones se crean de manera concéntrica alrededor de él. Si se determina que debe borrarse, el borrado será lógico, como en el caso anterior, para poder mantener los límites de todas las regiones.

En ninguna de las operaciones de borrado sería necesario realizar nuevos cálculos de la función de distancia.

En el caso de que el borrado afecte a un objeto que determina una región, se produce una situación de pérdida de precisión, en el sentido de que los límites de la región están representados por un objeto que ya no pertenece a la colección. Esto puede originar que en algún caso concreto se determine actuar sobre una región, por ser la más cercana al objeto de búsqueda, aunque esa cercanía venga dada por un objeto que realmente ya no es de la colección. Puede ocurrir que haya algún otro objeto relativamente cercano al que determina el límite de la región, pero en un caso límite podría no ser así. Por ejemplo, en la búsqueda kNN podría ocurrir que se determinase que una región es la más cercana al objeto de búsqueda, gracias a considerar la distancia a un objeto que delimita la región, y que a la vez dicho objeto ya no pertenezca al cluster, con lo que incluso podría ser realmente otra la región más cercana. La solución a este problema vendría dada por una reestructuración del índice, entendiendo por ello una actualización de sus datos. En cualquier caso, si el número de objetos borrados fuese importante sería necesario reestructurar el índice.

La estructura del índice elegida no facilita la implementación de las operaciones de borrado. La correspondencia posicional existente en el array de identificadores, determina a qué región pertenece un objeto mediante la ubicación de su identificador en el array, por lo que una operación de inserción necesita la reestructuración del índice. Aún así se perderían las referencias posicionales de los objetos que determinan las regiones, que habría que seguir manteniendo. Pueden darse algunas excepciones muy puntuales, como el caso de sustituir un elemento del array que había sufrido un borrado lógico por un nuevo elemento, pero en general, las inserciones necesitan la reestructuración del índice. Esto haría que la estrategia propuesta no fuese adecuada ante situaciones de inserción masivas.

Una alternativa que soluciona este problema, consiste en sustituir el array de identificadores de los objetos por un conjunto de arrays, uno para cada región del cluster. En este caso, cuando fuese necesario introducir un nuevo objeto en la colección, habría que calcular su distancia a los diferentes centros de los clusters, lo que determinaría tanto el cluster como la región a la que pertenece, con lo que su identificador se añadiría al array asociado a esa región. No sería necesario establecer

un orden dentro de cada array, ya que todos sus identificadores corresponden a objetos de la misma región.

Esta estructura alternativa mejora las prestaciones dinámicas de la estrategia. El espacio que ocupa es prácticamente igual al de la estructura inicial y en el caso del borrado, permite incluso que si un objeto no determina ninguna región, el borrado sea físico y no lógico. Además es adecuada para el tratamiento de inserciones masivas.

Pero a pesar de todas estas ventajas, si se produce un conjunto importante de modificaciones, ocurriría ineludiblemente un desequilibrio en el número de objetos que pertenecen a cada región, con lo que podría haber regiones con muchos objetos y otras con muy pocos, lo que ocasionaría una pérdida de rendimiento, como ya se ha mencionado en la sección 5.2. La solución a este problema sería también la reestructuración del índice.

5.7. Resumen

En este capítulo se ha presentado una nueva estrategia aplicable a los métodos de búsqueda por similitud basados en particiones, que utilizan el radio de cobertura en el criterio de descarte de los clusters. Se proponen algoritmos para la búsqueda por rango y para la búsqueda kNN que evitan la comparación exhaustiva del objeto de búsqueda, con los objetos de los clusters que no pueden descartarse con la información contenida en el índice, lo que supone la reducción de la complejidad externa. Los algoritmos se han implementado para el método Lista de Clusters, aunque son aplicables a los restantes métodos de las características citadas.

La nueva estrategia, denominada Reducción de Cluster, se basa en definir un número de regiones dentro de cada cluster, procurando que todas ellas tengan el mismo número de objetos. Al realizar la búsqueda en un cluster no descartado, se procesa cada una de las regiones, empezando por la más cercana al objeto de búsqueda, calculando la función de distancia para los objetos de la misma y después eliminando la región, originando una reducción progresiva del cluster al suprimir las regiones ya exploradas, hasta que puedan descartarse las regiones restantes. La definición de un reducido número de regiones dentro de cada cluster, reduce significativamente el coste de las búsquedas, originando un pequeño incremento del tamaño del índice y manteniendo la complejidad del espacio en $O(n)$.

Se ha realizado una evaluación experimental para la que se han utilizado colecciones tanto reales como sintéticas pertenecientes a la *Metric Spaces Library*, que muestran la mejora del rendimiento de las búsquedas, dependiendo de la colección y del número de regiones definidas, y el hecho de que si usamos un número

reducido de regiones, la mejora del rendimiento de las búsquedas es muy cercana a la que se obtiene al considerar que cada uno de los objetos del cluster define una región, que sería el caso de menor coste.

Finalmente se han discutido las capacidades dinámicas de la estrategia y cómo afectan a la estructura del índice.

Capítulo 6

Conclusiones

Desde los orígenes de la informática el volumen de datos almacenado en formato digital ha crecido progresivamente, pero ha sido en las dos últimas décadas cuando este hecho se ha intensificado, en parte gracias a que cada vez más se almacenan datos complejos que van desde ficheros con diversos grados de estructuración, a otros como documentos de texto, ficheros de sonido, de imágenes, de vídeo y otros que constituyen combinaciones de cualquiera de los anteriores y que a menudo tienen enlaces a otros documentos. La tradicional búsqueda exacta resulta inadecuada para aplicarla a estos datos, lo que conlleva que la búsqueda por similitud sea el objeto de interés de numerosas investigaciones y se convierta en una de las operaciones fundamentales en el campo de la Recuperación de la Información.

Una manera de implementar la búsqueda por similitud es mediante su formalización utilizando espacios métricos, ya que permiten determinar la similitud entre dos objetos mediante una función de distancia o métrica.

En esta tesis se abarca el problema de la búsqueda por proximidad usando espacios métricos, centrándose en plantear soluciones a la mejora del rendimiento general de las búsquedas. El estudio abarca los dos tipos de métodos existentes, los basados en pivotes y los basados en particiones, y se orienta en proporcionar mejoras en las características en las que cada uno de los métodos presenta debilidades.

En este capítulo se hace un resumen de las principales contribuciones de esta tesis y se mencionan líneas de trabajo futuro, algunas de las cuales constituyen problemas sobre los que actualmente estamos trabajando.

6.1. Contribuciones

En esta sección se describen las principales contribuciones de esta tesis.

Índice de pivote único (UPI)

Se ha presentado un nuevo método de búsqueda por similitud basado en pivotes denominado Índice de Pivote Único (UPI). En general los métodos basados en pivotes crean un índice en el que almacenan las distancias de unos cuantos objetos de referencia o pivotes, al resto de objetos de la base de datos, lo que conlleva que el tamaño ocupado por el índice suele ser muy voluminoso, hasta el punto de dificultar y a veces impedir su aplicación a casos reales en los que se realiza el tratamiento de grandes volúmenes de datos. UPI es un método basado en pivotes que se centra en reducir drásticamente el tamaño del índice asociado, a la vez que origina un número de evaluaciones de la función de distancia aceptable.

Para desarrollar el nuevo método se ha analizado la eficacia de los diversos pivotes para descartar cada uno de los objetos de la base de datos, se ha aplicado un método de selección de pivotes que garantiza que obtiene pivotes cercanos y lejanos a cada objeto, y se ha determinado, para cada colección analizada, los valores que hacen que el pivote más cercano y más lejano a un objeto sean eficientes para descartarlo, considerados en puntuaciones estándar. De esta forma se realiza una caracterización de las colecciones que permite establecer unos rangos de valores para los que esos pivotes producirán buenos resultados de cara al proceso de descartar cada objeto. El nuevo método consiste en almacenar en el índice, para cada objeto de la base de datos, la distancia a un único pivote, que es el más cercano o el más lejano al objeto, según los valores resultantes de sus puntuaciones estándar.

Utilizar un criterio de selección de pivotes que garantice su buena distribución en el espacio y la elección del pivote más prometedor para cada objeto de la base de datos siguiendo el procedimiento de las puntuaciones estándar, aportan un carácter riguroso al método que hace que el pivote elegido tenga realmente muchas posibilidades de descartar ese objeto.

Los resultados experimentales muestran que en la reducción del espacio del índice, UPI mejora de manera importante a otros métodos basados en pivotes, incluso a alguno que tenía también ese objetivo, y además genera también índices de menor tamaño que los obtenidos por un método representativo basado en particiones, como es Lista de Clusters, mostrando un comportamiento inusual en los métodos basados en pivotes, que siempre requieren más espacio que los basados en particiones. Mantiene además el número de evaluaciones de la función de distancia

en valores aceptables, ya que en este indicador también supera, en casi todos los casos estudiados, a Lista de Clusters.

Las consecuencias inmediatas derivadas de las características de UPI son varias, entre ellas que puede aplicarse en escenarios reales que otros métodos basados en pivotes no podían abordar debido al problema del tamaño del índice, que resalta la importancia de los factores relacionados con el espacio ocupado por el índice en el coste global de una consulta, como son el tiempo de entrada/salida y el tiempo de CPU necesario para procesar la estructura del índice, factores que a veces otros métodos no tienen en cuenta, y que su rendimiento lo convierte en una alternativa competitiva frente a los métodos basados en particiones, aunque estos últimos presentan unas características muy diferentes.

Reducción de Cluster (CR)

Se ha introducido una nueva estrategia aplicable a los métodos de búsqueda por similitud basados en particiones denominada Reducción de Cluster (CR). Los métodos basados en particiones utilizan los centros de los clusters como objetos de referencia, haciendo que en general cada cluster esté formado por los objetos que están más cerca de su centro que de cualquier otro. El proceso de descarte se realiza sobre la totalidad de cada cluster, siguiendo uno de los dos criterios existentes.

En uno de los criterios se utiliza la distancia máxima del centro a los objetos del cluster, denominada radio de cobertura, almacenando en el índice esa distancia y la referencia del centro del cluster. Dado un objeto de búsqueda, para cada cluster se calcula la distancia del objeto a su centro y junto a los datos del índice, se determina si algún objeto del cluster puede cumplir o no el criterio de búsqueda. En el caso negativo se descarta la totalidad del cluster sin comparar el objeto de búsqueda con sus objetos. En el caso afirmativo hay que comparar el objeto de búsqueda con todos los objetos del cluster para determinar cuál de ellos cumple el criterio. Este proceso de comprobación exhaustiva de los clusters que no resultan descartados, es un punto débil de estos métodos, ya que genera muchas comparaciones infructuosas y hace que la elección de los centros y de los radios de cobertura tenga mucha influencia en la eficiencia de los métodos.

En esta tesis se ha propuesto una nueva estrategia que tiene como objetivo reducir el número de comparaciones de la función de distancia en los clusters que no han resultado descartados. Consiste en definir un número de regiones dentro de cada cluster, procurando que todas tengan el mismo número de elementos. Ante una búsqueda en la que un cluster no resulta descartado, en vez de explorarlo exhaustivamente calculando la distancia del objeto de búsqueda a cada objeto del cluster, con esta nueva estrategia se restringe la exploración a las regiones del cluster, empezando por la más cercana al objeto de búsqueda y eliminándola una vez

comprobados sus elementos, de forma que el cluster sufre una reducción progresiva, llegando un momento en el que el resto de sus regiones pueden descartarse sin comparar sus objetos con el objeto de búsqueda. Esto origina una importante reducción del número de evaluaciones de la función de distancia, que es el factor en el que estos métodos obtienen resultados menos destacables.

La propuesta incluye los algoritmos de búsqueda por rango y de búsqueda kNN necesarios para su aplicación. Aunque se modifica el índice, ya que se almacenan los objetos que delimitan las regiones y sus distancias al centro del cluster, además de los identificadores de los objetos que pertenecen a cada región, y los algoritmos de búsqueda cambian sustancialmente con respecto a los existentes, hablamos de una nueva estrategia y no de un nuevo método, puesto que entendemos que no requiere un nuevo índice, sino que solo con unas pequeñas modificaciones en los existentes, se podría aplicar a cualquier método basado en particiones que use el radio de cobertura como criterio de descarte. Esto no solo no lo consideramos una limitación, sino que es una importante ventaja de nuestra propuesta, ya que utilizando la nueva estrategia, los métodos existentes mejorarán de manera importante, tal como ha ocurrido en la experimentación realizada sobre el método Lista de Clusters, donde tanto las búsquedas por rango como las kNN, reducen significativamente el número de evaluaciones de la función de distancia.

6.2. Trabajo futuro

En esta sección se plantean algunas líneas de actuación para futuros trabajos relacionados con esta tesis.

- En lo referente a los métodos basados en pivotes, en el desarrollo de este trabajo ha surgido la idea de buscar alternativas que permitan agilizar la elección de los pivotes, combinando criterios heurísticos con formales, de forma que permitan seleccionar pivotes que tengan posibilidades de descartar cada objeto de la base de datos, como hace UPI, pero que el proceso de elección sea mucho más rápido y no requiera analizar las características de la colección. Aunque la óptica cambia radicalmente con relación a la propuesta de esta tesis, se mantienen los objetivos de reducir el espacio del índice y de lograr un número de evaluaciones de la función de distancia competitivo con los métodos basados en particiones, de la misma forma que se planteó en UPI, porque nos parecen logros destacables que amplían las posibilidades de aplicación de los métodos a casos reales. Esta línea se encuentra en una fase inicial de la investigación, aunque ya se están realizando pruebas preliminares de alguna alternativa.

- Respecto a los métodos basados en particiones, la idea es ampliar las posibilidades de Reducción de Cluster para lograr un proceso de exploración de las regiones que abarque menos objetos que los que poseen, sin incrementar el tamaño de los índices ni generar más comparaciones de la función de distancia. Este trabajo se encuentra también en una fase preliminar, aunque ya se han realizado algunas propuestas que inicialmente no dieron resultados muy positivos, pero que su revisión se encamina hacia modificaciones que parecen ser más prometedoras.

- Una idea que ha surgido reiteradamente a lo largo de la investigación que ha dado lugar a esta tesis, es la de explorar técnicas que vayan más allá de las propiedades de los espacios métricos, neutralizando el efecto de alguna de ellas, e incluso evitándolo. Esta idea entronca con una línea de investigación de búsqueda por similitud muy reciente, que trata globalmente de búsquedas en espacios no métricos. Esta línea de actuación es por ahora un proyecto, aunque se espera afrontarlo de inmediato.

- Finalmente hay un importante campo de investigación en el que deseamos adentrarnos, que es el de la aplicación efectiva de los métodos de búsqueda por similitud a problemas reales, concretos y delimitados. Es especial consideramos que los procesos de análisis y de decisión que se presentan en diferentes dominios, pueden ser susceptibles de aplicación de métodos de búsqueda por similitud, tanto genéricos como orientados a sus características específicas. Este trabajo es un proyecto a medio plazo.

Apéndice A

Notación utilizada

En la tabla siguiente se resume la notación utilizada:

Símbolo	Significado
X	Universo de objetos válidos
(X, d)	Espacio métrico
d	Función de distancia, métrica o distancia
$\mathbb{R}$	Conjunto de los números reales
U	Colección de objetos o base de datos
n	Tamaño de la base de datos
$R(q, r)$	Búsqueda por rango
q	Objeto de consulta o de búsqueda
r	Radio de la búsqueda por rango
$kNN(q)$	Búsqueda de los k-vecinos más cercanos
L_p	Familia de distancias de Minkowski
$\rho = \mu^2/2\sigma^2$	Estimación de la dimensionalidad intrínseca
P	Conjunto de pivotes
m	Número de elementos del conjunto de pivotes. Mediana de las distancias de un pivote a los restantes objetos (VPT).
C	Una de las zona que forman una partición del espacio
C_i	Zona i-ésima de las que forman una partición del espacio
c_i	Centro de una zona C_i
r_c	Radio de cobertura o mayor distancia del centro a otro objeto de la misma zona

Apéndice B

Lista de acrónimos

En la tabla siguiente se describen los acrónimos utilizados:

Acrónimo	**Significado**
AESA	Approximating and Eliminating Search Algorithm
BKT	Burkhard-Keller Tree
BST	Bisector Tree
CR	Reducción de Cluster (Cluster Reduction)
FHFQT	Fixed Height Fixed Queries Tree
FQA	Fixed Queries Array
FQT	Fixed Queries Tree
GHT	Generalized Hyperplane Tree
GNAT	Geometric Near-neighbor Access Tree
KVP	K Vantage Points
LAESA	Lineal AESA
LC	Lista de Clusters
MT	M-Tree
MVPT	Multi Vantage Point Tree
UPI	Índice de Pivote Único (Unique Pivot Index)
VPT	Vantage Point Tree
VT	Voronoi Tree
SAT	Spatial Approximation Tree
SSS	Sparse Spatial Selection

Apéndice C

Publicaciones

En este apéndice se mencionan las publicaciones directamente relacionadas con esta tesis, indicando otras publicaciones que las han citado y menciones que se les han concedido, con los datos disponibles en el mes de junio de 2012.

Congresos internacionales

- Ares, L. G.; Brisaboa, N. R.; Ordóñez, A.; Pedreira, O. (2012). «Efficient Similarity Search in Metric Spaces with Cluster Reduction». Artículo aceptado en: *5th International Conference on Similarity Search and Applications (SISAP 2012)*, Toronto, Canada.

- Ares, L. G.; Brisaboa, N. R.; Esteller, M. F.; Pedreira, O.; Places, A. S. (2009). «Optimal Pivots to Minimize the Index Size for Metric Access Methods». En: *Proc. of the 2nd International Workshop on Similarity Search and Applications (SISAP 2009)*, pp. 74-80, Prague, Czech Republic. IEEE Press.

  Este artículo ha sido citado por:

  – Socorro, R.; Micó, L.; Oncina, J. (2011). «A fast pivot-based indexing algorithm for metric spaces». *Pattern Recognition Letters*, 32(11), pp. 1511-1516.

Congresos nacionales

- Ares, L. G.; Brisaboa, N. R.; Ordóñez, A.; Pedreira, O. (2012). «Reducción de la Complejidad Externa en Búsquedas por Similitud usando Técnicas de Clustering». Artículo aceptado en: *XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2012)*.

- Ares, L. G.; Brisaboa, N. R.; Bustos, B.; Ordóñez, A.; Pedreira, O. (2010). «Optimización de las Búsquedas kNN en Espacios Métricos». En: *Actas de las XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010)*, pp. 153-162, Valencia, España.

- Ares, L. G.; Brisaboa, N. R.; Esteller, M. F.; Pedreira, O.; Places, A. S. (2009). «Reducción de Tamaño del Índice en Búsquedas por Similitud sobre Espacios Métricos». En: *Actas de las XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2009)*, pp. 249-260, San Sebastián, España.

  Este trabajo recibió el premio al mejor artículo de la conferencia.

Revistas nacionales

- Ares, L. G.; Brisaboa, N. R.; Esteller, M. F.; Pedreira, O.; Places, A. S. (2010). «Reducción de Tamaño del Índice en Búsquedas por Similitud sobre Espacios Métricos». En: *Novática, 204*, pp. 50-54, ATI, Madrid, España.

Apéndice D

Nota sobre las referencias históricas

En diferentes puntos del presente trabajo se han incluido comentarios históricos y hasta en un caso, una anécdota. La justificación de que tengan cabida en un trabajo de este tipo es doble; por una parte se han incluido únicamente cuando presentaban una rigorosa constatación, avalada por varias fuentes solventes que además tuviesen diferentes orígenes,[1] y por otra, al intento de reflejar la necesidad de enmarcar las ideas y los conocimientos científicos, en la época histórica y en las circunstancias sociales en las que se desarrollaron, para entender mejor el proceso que ha llevado a su génesis, además de otorgar una consideración plenamente justificada a las personas que las generaron y las desarrollaron.

[1] Evidentemente la excepción aquí la constituye la "mosca de Descartes", de la que desgraciadamente no hay referencias históricas concluyentes, porque en otro caso, seguramente sería un buen motivo en el que apoyarse para enfocar el estudio de la Geometría en las primeras etapas escolares de una manera amena y divertida, algo siempre necesario ante la aridez de los conceptos matemáticos. De todas formas, la creencia de que la inspiración del sistema de coordenadas se produjo gracias a la famosa mosca, está ampliamente aceptada, ya que aparece en numerosas referencias y concuerda también con las costumbres documentadas acerca de la vida de Descartes, por lo que se ha incluido aquí a modo de anécdota.

Bibliografía

Ares, Luis G.; Brisaboa, Nieves R.; Bustos, Benjamín; Ordóñez, Alberto y Pedreira, Óscar (2010). «Optimización de las Búsquedas kNN en Espacios Métricos». En: *Actas de las XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010)*, pp. 153–162. Valencia. [Pág.: 128]

Baeza-Yates, Ricardo (1997). «Searching: an algorithmic tour». *Encyclopedia of Computer Science and Technology*, 37, pp. 331–359. [Pág.: 65]

Baeza-Yates, Ricardo; Cunto, Walter; Manber, Udi y Wu, Sun (1994). «Proximity matching using fixed-queries trees». En: *Proc. of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM 1994)*, pp. 198–212. Springer-Verlag, Asilomar, C.A.. [Pág.: 62]

Beyer, Kevin; Goldstein, Jonathan; Ramakrishnan, Raghu y Shaft, Uri (1999). «When Is "Nearest Neighbor" Meaningful?» En: *In Int. Conf. on Database Theory*, pp. 217–235. [Pág.: 52]

Bille, Philip (2005). «A survey on tree edit distance and related problems». *Theoretical Computer Science*, 337(1-3), pp. 217–239. [Pág.: 21]

Böhm, Christian; Berchtold, Stefan y Keim, Daniel A. (2001). «Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases». *ACM Comput. Surv.*, 33, pp. 322–373. [Pág.: 52]

Bozkaya, Tolga y Ozsoyoglu, Meral (1997). «Distance-based indexing for high-dimensional metric spaces». En: ACM Press (Ed.), *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1997)*, pp. 357–368. Tucson, Arizona, USA. [Pág.: 71, 72]

Brin, Sergey (1995). «Near neighbor search in large metric spaces». En: *Proc. of 21st conference on Very Large Databases (VLDB 1995)*, ACM Press. [Pág.: 81, 105]

Brisaboa, Nieves R.; Fariña, Antonio; Pedreira, Óscar y Reyes, Nora (2006). «Similarity search using sparse pivots for efficient multimedia information retrieval». En: *Proc. of the 8th IEEE International Symposium on Multimedia (ISM 2006)*, pp. 881–888. IEEE Press, San Diego, California, USA. [Pág.: 90, 102, 104, 106]

Burkhard, Walter A. y Keller, Robert M. (1973). «Some approaches to best-match file searching». *Communications of the ACM*, 16(4), pp. 230–236. ACM Press. [Pág.: 59]

Bustos, Benjamín y Morales, Nelson (2010). «On the asymptotic behavior of nearest neighbor search using pivot-based indexes». En: *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP 2010, pp. 33–39. ACM, New York, NY, USA. [Pág.: 94]

Bustos, Benjamín; Navarro, Gonzalo y Chávez, Edgar (2003). «Pivot selection techniques for proximity searching in metric spaces». *Pattern Recognition Letters*, 24(14), pp. 2357–2366. Elsevier. [Pág.: 102, 106, 115, 118]

Bustos, Benjamín; Pedreira, Óscar y Brisaboa, Nieves R. (2008). «A dynamic pivot selection technique for similarity search in metric spaces». En: *Proc. of 1st Int. Workshop on Similarity Search and Applications (SISAP 2008)*, pp. 105–112. IEEE Press. [Pág.: 102, 104, 106]

Celik, Cengiz (2002). «Priority Vantage Points Structures for Similarity Queries in Metric Spaces». En: *Proc. of EurAsia-ICT 2002: Information and Communication Technology*, volumen 2510 de *Lecture Notes in Computer Science*. Springer. [Pág.: 109, 128]

Chávez, Edgar; Marroquín, José Luis y Navarro, Gonzalo (1999). «Overcoming the curse of dimensionality». En: *European Workshop on Content-based Multimedia Indexing (CBMI 1999)*, pp. 57–64. [Pág.: 67]

Chávez, Edgar; Marroquín, José Luis y Navarro, Gonzalo (2001a). «Fixed Queries Array: A fast and economical data structure for proximity searching». *Multimedia Tools and Applications*, 14(2), pp. 113–135. [Pág.: 66, 67]

Chávez, Edgar y Navarro, Gonzalo (2000). «An Effective Clustering Algorithm to Index High Dimensional Metric Spaces». En: *Proc. 7th International Symposium on String Processing and Information Retrieval (SPIRE 2000)*, pp. 75–86. IEEE CS Press. [Pág.: 87]

Chávez, Edgar y Navarro, Gonzalo (2005). «A Compact Space Decomposition for Effective Metric Indexing». *Pattern Recognition Letters*, 26(9), pp. 1363–1376. [Pág.: 88, 137]

Chávez, Edgar; Navarro, Gonzalo; Baeza-Yates, Ricardo y Marroquín, José Luis (2001b). «Searching in metric spaces». *ACM Computing Surveys*, 33(3), pp. 273–321. ACM Press. [Pág.: 3, 25, 33, 38, 52, 56, 99, 102, 104]

Ciaccia, Paolo; Patella, Marco y Zezula, Pavel (1997). «M-tree: An Efficient Access Method for Similarity Search in Metric Spaces». En: *Proc. of the 23rd International Conference on Very Large Data Bases (VLDB 1997)*, pp. 426–435. ACM Press, Athens, Greece. [Pág.: 84]

Ciaccia, Paolo; Patella, Marco y Zezula, Pavel (1998). «A cost model for similarity queries in metric spaces». En: *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS 1998, pp. 59–68. ACM, New York, NY, USA. [Pág.: 94]

de la Higuera, Colin y Micó, Luisa (2008). «A contextual normalized edit distance». En: *Proceedings of First International Workshop on Similarity Search and Applications (SISAP 2008)*, pp. 61–68. IEEE Press, Cancún, México. [Pág.: 21]

Dehne, Frank y Noltemeier, Hartmut (1987). «Voronoi trees and clustering problems». *Information Systems*, 12(2), pp. 171–175. [Pág.: 79]

Faloutsos, Christos; Barber, Ron; Flickner, Myron; Hafner, Jim; Niblack, Wayne; Petkovic, Dragutin y Equitz, William (1994). «Efficient and Effective Querying by Image Content». *Journal of Intelligent Information Systems*, 3(3/4), pp. 231–262. [Pág.: 20]

Faragó, A.; Linder, T. y Lugosi, G. (1993). «Fast Nearest-Neighbor Search in Dissimilarity Spaces». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, pp. 957–962. [Pág.: 102]

Figueroa, Karina (2007). *Indexación efectiva de espacios métricos usando permutaciones*. Tesis doctoral, Departmento de Ciencias de la Computación, Universidad de Chile. [Pág.: 56]

Figueroa, Karina; Chávez, Edgar; Navarro, Gonzalo y Paredes, Rodrigo (2009). «Speeding up Spatial Approximation Search in Metric Spaces». *ACM Journal of Experimental Algorithmics (JEA)*, 14, p. article 3.6. [Pág.: 101]

Guha, Sudipto; Jagadish, H. V.; Koudas, Nick; Srivastava, Divesh y Yu, Ting (2002). «Approximate XML joins». En: *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD 2002)*, pp. 287–298. ACM Press, Madison, Wisconsin. [Pág.: 21]

Hafner, James; Sawhney, Harpreet S.; Equitz, Will; Flickner, Myron y Niblack, Wayne (1995). «Efficient color histogram indexing for quadratic form distance functions». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7), pp. 729–736. [Pág.: 20]

Hennig, Christian y Latecki, Longin Jan (2003). «The choice of vantage objects for image retrieval». *Pattern Recognition*, 36(9), pp. 2187–2196. [Pág.: 106]

Hjaltason, Gisli R. y Samet, Hanan (2003). «Index-Driven Similarity Search in Metric Spaces». *ACM Transactions on Database Systems (TODS 2003), ACM Press*, 28(4), pp. 517–580. [Pág.: 33]

Huttenlocher, Daniel P.; Klanderman, Gregory A. y Rucklidge, William (1993). «Comparing Images Using the Hausdorff Distance». *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9), pp. 850–863. [Pág.: 23]

Ibáñez, Raúl (2010). *La cuarta dimensión*. RBA. [Pág.: 13]

Ioka, Mikihiro (1989). «A method of defining the similarity of images on the basis of color information». *Technical report RT-0030*, IBM Tokyo Research Lab. [Pág.: 20]

Kalantari, Iraj y McDonald, Gerard (1983). «A data structure and an algorithm for the nearest point problem». *IEEE Transactions on Software Engineering*, 9, pp. 631–634. IEEE Press. [Pág.: 77]

Korn, Flip y Muthukrishnan, S. (2000). «Influence sets based on reverse nearest neighbor queries». En: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD 2000, pp. 201–212. ACM, New York, NY, USA. ISBN 1-58113-217-4. [Pág.: 29]

Levenshtein, Vladimir (1965). «Binary codes capable of correcting spurious insertions or deletions of ones». *Problems of Information Transmission*, 1, pp. 8–17. [Pág.: 21]

Micó, Luisa; Oncina, José y Vidal, R. Enrique (1994). «A new version of the nearest-neighbor approximating and eliminating search (AESA) with linear pre-processing time and memory requirements». *Pattern Recognition Letters*, 15, pp. 9–17. Elsevier. [Pág.: 75, 102, 105, 106]

Navarro, Gonzalo (1999). «Searching in metric spaces by spatial approximation». En: *Proceedings of String Processing and Information Retrieval (SPIRE 1999)*, pp. 141–148. IEEE Computer Science Press. [Pág.: 85]

Navarro, Gonzalo (2009). «Analyzing Metric Space Indexes: What For?» En: *Proceedings of the 2009 Second International Workshop on Similarity Search and Applications*, SISAP 2009, pp. 3–10. IEEE Computer Society, Washington, DC, USA. [Pág.: 94]

Navarro, Gonzalo y Reyes, Nora (2002). «Fully Dynamic Spatial Approximation Trees». En: *Proc. 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, LNCS 2476, pp. 254–270. Springer. [Pág.: 85]

Noltemeier, Hartmut (1989). «Voronoi trees and applications». En: *Proceedings of the International Workshop on Discrete Algorithms and Complexity*, pp. 69–74. Japan. [Pág.: 79]

Novak, David y Batko, Michal (2009). «Metric Index: An Efficient and Scalable Solution for Similarity Search». En: *Proceedings of the Second International Workshop on Similarity Search and Applications*, SISAP 2009, pp. 65–73. IEEE Computer Society. [Pág.: 94]

Novak, David; Batko, Michal y Zezula, Pavel (2011). «Metric Index: An efficient and scalable solution for precise and approximate similarity search». *Information Systems*, 36(4), pp. 721–733. [Pág.: 94]

Patella, Marco (1999). *Similarity Search in Multimedia Databases.* Tesis doctoral, Department of Electronics, Computer Sciences and Systems, University of Bologna. [Pág.: 31]

Pedreira, Óscar (2009). *An effective approach for selecting indexing objects in metric spaces.* Tesis doctoral, Departmento de Computación, Universidade da Coruña. [Pág.: 56]

Rosenfeld, B.A. (1988). *A History of Non-Euclidean Geometry.* Springer-Verlag. [Pág.: 12]

Samet, Hanan (2006). *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling).* Morgan Kaufmann Publishers Inc.. [Pág.: 33, 86]

Sankoff, D. y Kruskal, J.B. (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley. [Pág.: 21]

Santini, Simone y Jain, Ramesh (1999). «Similarity Measures». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9), pp. 871–883. [Pág.: 25]

Searcóid, Mícheál Ó (2007). *Metric Spaces.* Springer Undergraduate Mathematics Series. Springer. [Pág.: 16]

Shaft, Uri y Ramakrishnan, Raghu (2006). «Theory of nearest neighbors indexability». *ACM Trans. Database Syst.*, 31, pp. 814–838. [Pág.: 52]

Skopal, Tomás; Pokorný, Jaroslav y Snásel, Václav (2004). «PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases». En: *ADBIS (Local Proceedings)*, . [Pág.: 93]

Skopal, Tomáš (2010). «Where are you heading, metric access methods?: a provocative survey». En: *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP 2010, pp. 13–21. ACM, New York, NY, USA. [Pág.: 25, 101]

Skopal, Tomáš y Bustos, Benjamín (2009). «On index-free similarity search in metric spaces». En: *Proc. of 20th Conf. on Database and Expert Systems Applications (DEXA 2009)*, LNCS(5690), pp. 516–531. Springer. [Pág.: 98]

Socorro, Raisa; Micó, Luisa y Oncina, José (2011). «A fast pivot-based indexing algorithm for metric spaces». *Pattern Recognition Letters*, 32(11), pp. 1511 – 1516. [Pág.: 101]

Traina Jr., Caetano; Traina, Agma J. M.; Faloutsos, Christos y Seeger, Bernhard (2002). «Fast Indexing and Visualization of Metric Data Sets using Slim-Trees». *IEEE Trans. Knowl. Data Eng.*, 14(2), pp. 244–260. [Pág.: 93]

Traina Jr., Caetano; Traina, Agma J. M.; Seeger, Bernhard y Faloutsos, Christos (2000). «Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes». En: *EDBT*, pp. 51–65. [Pág.: 93]

Uhlmann, Jeffrey K. (1991). «Satisfying general proximity/similarity queries with metric trees». *Information Processing Letters*, 40, pp. 175–179. Elsevier. [Pág.: 67, 80]

van Leuken, Reinier H.; Veltkamp, Remco C. y Typke, Rainer (2006). «Selecting vantage objects for similarity indexing». En: *Proc. of the 18th International Conference on Pattern Recognition (ICPR 2006)*, pp. 453–456. IEEE Press. [Pág.: 106]

Venkateswaran, Jayendra; Kahveci, Tamer; Jermaine, Christopher M. y Lachwani, Deepak (2008). «Reference-based indexing for metric spaces with costly distance measures». *The VLDB Journal*, 17(5), pp. 1231–1251. Springer. [Pág.: 106]

Vidal, Enrique (1986). «An algorithm for finding nearest neighbors in (approximately) constant average time». *Pattern Recognition Letters*, 4, pp. 145–157. Elsevier. [Pág.: 72]

Vidal, Enrique (1994). «New formulation and improvements of the nearest-neighbor approximating and eliminating search algorithm (AESA)». *Pattern Recognition Letters*, 15(1), pp. 1–7. [Pág.: 72, 74]

Vleugels, Jules y Veltkamp, Remco C. (2002). «Efficient image retrieval through vantage objects». *Pattern Recognition*, 35(1), pp. 69–80. Elsevier. [Pág.: 106]

Yianilos, Peter (1993). «Data structures and algorithms for nearest-neighbor search in general metric spaces». En: *Proc. of the fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1993)*, pp. 311–321. ACM Press. [Pág.: 67, 70, 105]

Zezula, Pavel; Amato, Giuseppe; Dohnal, Vlatislav y Batko, Michal (2006). *Similarity search. The metric space approach*. Advances in Database Systems. Springer. [Pág.: 33, 38, 57]

Zhang, K. y Shasha, D. (1989). «Simple fast algorithms for the editing distance between trees and related problems». *SIAM J. Computing*, 18(6), pp. 1245–1262. [Pág.: 21]