Abstract

This research is presented to give conformity to the requirements demanded by the University of the Bio Bio in the qualification process for the Civil Engineering degree in Computer Science. The thesis, entitled "Compact Data Structures for Indexing Points of n Dimensions and Their Applications," addresses problem of storing large amounts of data in a compact way without losing the ability to respond to certain queries.

Compact data structures are useful when storing a large number of points, for example: vehicles and buildings in a city, because they minimize the use of space, storing the minimum information of the data, reaching this information to be stored in memory RAM, and making it much faster. At the same time they are able to respond to a series of queries to manipulate the data without decompressing the structure.

This paper performs a systematic review of the literature of compact data structures that respond to queries such as proximity and rank amongst others – it was done by searching for keywords, then documents were selected by inclusion criteria, recording the results.

From the results of the search, a data structure was selected, which was available by the author, and experiments were carried out recording the size of the data structure and the time used to carry out the query by range with different ticket data. Finally, the conclusions that were able to be obtained from this investigation will be presented.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la actualidad, una infinidad de situaciones se pueden representar mediante puntos (coordenadas en dos dimensiones), por ejemplo vehículos en la ciudad, barcos en el mar, edificios en la ciudad por lo que resulta de gran importancia tener una estructura de datos que pueda almacenarla, y realizar consultas como por ejemplo búsqueda de rango ortogonal, en segundo lugar locación de puntos y para finalizar la familia de puntos de vecinos más cercanos ya sea el vecino, el par de vecinos más cercanos, etc.

Por ejemplo, si tenemos una base de datos de imágenes médicas con tumores, es posible, dada la imagen de un órgano de un paciente, encontrar la imagen más similar dentro de la base de datos y así determinar si existe alguna probabilidad de encontrar un tumor en dicho órgano, mucho más rápido que de forma manual. Otra área de aplicación es en la astronomía donde diariamente se procesan y almacenan grandes volúmenes de datos, muchos de ellos corresponden a imágenes multi-espectrales del universo. Dichas imágenes pueden ser representadas como un conjunto de puntos de $n \geq 1$ dimensiones.

Al igual que en astronomía, en muchas áreas de aplicación se generan grandes volúmenes de información. Por esta razón, contar con estructuras de datos que minimicen el uso del espacio y que tengan algoritmos eficientes es importante. Una manera de lograr lo anterior es mediante las llamadas estructuras de datos compactas.

Al minimizar el espacio que ocupa la estructura de datos compacta es posible contener

toda la base de datos en la memoria RAM la cual es 6 órdenes de magnitud más rápida que el disco.

La estructuras de datos en dos dimensiones, por ejemplo, los wavelet-trees requieren un espacio cercano al peor caso de la entropía para almacenar puntos y logran complejidades de tiempo cercanas a las mejores. Sin embargo, todas las estructuras de datos conocidas que logran tiempos de consulta poli-logaritmicos en una dimensión mayor a 2, requieren espacio super lineal ($\mathcal{O}(n \log^{d-1} n)$ bits). Por lo tanto, los wavelet-trees son menos interesantes para grandes dimensiones, porque de forma muy rápida aumentan su tamaño para almacenar los datos.

Sin embargo, algunas consultas restringidas pueden hacerse en una dimensión adicional sin aumentar de gran manera el espacio, por ejemplo, dando pesos a los puntos, la consulta que podrá responder es buscar los puntos más pesados en un rango que puede interpretarse como una tercera dimensión (el peso) y realizando consultas de rango ordenado en el área.

Los quad-tree, en cambio, puede extenderse a d dimensiones sin exceder los $\mathcal{O}(dn \log n)$ bits, donde cada nodo divide a la mitad el espacio de coordenadas a través de cada dimensión, y luego tiene 2^d hijos. El árbol k^2-tree de la misma forma puede extenderse al k^d-tree de d dimensiones.

Por lo anterior en esta memoria de título se abordó el tema de las estructura de datos compactas para la indexación de puntos en n dimensiones y sus aplicaciones.

1.1. Objetivos

Los objetivos de la investigación es indicar cuales son los propósitos de esta, y nos permiten dejar indicado de forma clara cual es el alcance nuestro trabajo y definir de manera específica lo que se aspira a lograr. Estos objetivos se dividen en objetivo general y objetivos específicos.

1.1.1. Objetivo general

El objetivo general de esta investigación es examinar la literatura referente a la indexación y consulta de puntos en un espacio de n dimensiones que utilicen estructura de datos compactas con la finalidad de comparar dichas estructuras.

1.1.2. Objetivos Específicos

Para lograr el objetivo mencionado anteriormente se destacan tres objetivos específicos que nos ayudan a completar el objetivo general. Estos son:

- Seleccionar los artículos más relevantes siguiendo la metodología de la revisión sistemática de la literatura
- Comparar teóricamente los artículos seleccionados basado en las consultas soportadas, algoritmos y su análisis
- Implementar una de las estructuras para estudiar como se comporta en la práctica.

1.2. Organización del documento

A continuación se detallan las secciones y/o capítulos en las que se ha dividido el documento.

En el capítulo uno, se presenta los planteamientos y la justificación del trabajo, donde se describe la motivación que originó la realización de este trabajo, así como también destacar las áreas en las que aporte conocimiento.

De forma posterior, en el segundo capítulo, se presentan definiciones o conceptos previos que serán utilizados en durante el desarrollo del documento.

En el capítulo tercero se detalla la metodología utilizada para la realización de esta búsqueda. De igual forma se describe las palabras claves utilizada para realizar la búsqueda en la literatura.

En el capítulo cuarto se define el estado del arte y se muestra los resultados de la investigación o búsqueda realizada.

En el quinto capítulo se representan los resultados de la experimentación de la estructura de datos.

Finalmente, en el capítulo seis se plantean ideas de trabajos futuros relacionados con las

estructura de datos compactas, se presentan las conclusiones obtenidas y se sintetiza la totalidad de los planteamientos realizados en el desarrollo del presente trabajo.

Capítulo 2

Conceptos Previos

En el presente capítulo se presentan los conceptos claves presentes durante el desarrollo de este documento y que, por lo tanto, son fundamentales para comprender este trabajo.

2.1. Secuencia binaria

En ciencias de la computación una secuencia binaria es una cadena de números usando un sistema de numeración donde estos solamente se representan utilizando dos cifras: ceros y unos.

Sobre las secuancias binarias, además de saber el valor de un bit en particular, es posible realizar dos tipos de consultas estas son (Jacobson, 1989):

- $\mathsf{rank_b}$(B,i): responde la cantidad de bits en la secuencia binaria [1,i] con valor igual a b.
- $\mathsf{select_b}$(B,i): responde la posición de la ocurrencia del i-ésimo bit con valor b en la secuencia binaria llamada B.

Por ejemplo si $B = \{1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0\}$, la consulta $\mathsf{select_1}$(B,5) devuelve como valor 8 y $\mathsf{rank_1}$(B,5) devuelve como valor 3.

2.2. Entropía en teoría de la información

Uno de los grandes aportes en relación a este tema lo realizó E. Shannon (1948) que fue en su teoría matemática de la comunicación donde ésta propuso una manera de medir la cantidad de información. Define la cantidad de información en bits de un símbolo x como:

$$h(x) = \log_2(\frac{1}{P(x)}) \tag{2.1}$$

Donde $P(x)$ es la probabilidad de ocurrencia de x.

Dado un $P(x)$ cercano a uno se observa que $h(x)$ es cercano a 0, ya que no aportaría gran información, por ejemplo, si tenemos una moneda donde sus los lados son cara, al lanzar una moneda no aportaría información, dado que ya sabemos el resultado que se obtendrá. De forma inversa si la probabilidad se acerca a 0 la cantidad que información que aporta x es alta.

Junto con el concepto de la cantidad de información que aporta un carácter y/o símbolo, se puede saber la cantidad de información que aporta en promedio un vocabulario lo que se denomina entropía.

Dado un vocabulario X=$\{x_1, x_2, ..., x_i\}$, la entropía de vocabulario H(X) se define como (Navarro, 2016):

$$H(X) = \sum_{i=1}^{n} P(x_i) \log_2(\frac{1}{P(x_i)}) \tag{2.2}$$

Junto con E. Shannon (1948), Navarro (2016) define la entropía en términos generales como el número de bits necesarios para identificar sin ambigüedades un objeto dentro de un conjunto, este concepto también se puede aplicar a estructura de datos y conseguir el número mínimo de bits necesario para que se almacene la información en dicha estructura de datos.

2.3. Consultas espaciales

Al igual que en una base de datos tradicional, es posible recuperar información de una bases de datos espacial por medio de consultas. Las consultas espaciales se pueden clasificar en tres tipos, de rango, ubicación de puntos y basadas en distancia. A continuación se explicarán con más detalle.

2.3.1. Búsqueda de rango ortogonal

La consulta de búsqueda de rango ortogonal consiste en (He, 2013): Dado un rango p de consulta alineado con el eje dimensional d, se obtienen todos los puntos q que tienen al menos un punto en común con p. Formalmente se define como $RQ(p) = \{q \,|\, p \bigcap q \neq \emptyset\}$. Por ejemplo dada mi ubicación actual encontrar todos los locales de comida rápida que se encuentran dentro de un rango de consulta.

2.3.2. Locación de puntos

La locación de puntos es una consulta que consiste en (He, 2013): Dado un punto p, la consulta obtiene todos los puntos q que cubren a p. Formalmente se define $PQ(p) = \{q \,|\, p \bigcap q = p\}$. Por ejemplo obtener todas las farmacias que se encuentren en un determinado lugar.

2.3.3. Vecino(s) más cercano(s)

Su objetivo consiste en dado un punto p encontrar todos los puntos q que tienen la mínima distancia a p. Formalmente se define como $NN(p) = \{p \,|\, \forall q' : \text{dist}(p, q) \leq \text{dist}(p, q')\}$. En este caso *dist* se define como la distancia euclídea entre puntos. Por ejemplo, dada mi ubicación actual encontrar el cine más cercano.

Una extensión de este problema consiste en encontrar los k vecinos más cercanos. Junto con existen otras consultas basadas en proximidad como los vecinos reversos que consiste en encontrar los puntos de los cuales un punto p es el vecino más cercano.

Capítulo 3

Metodología

Para una revisión completa de la literatura de un tópico, es de gran importancia planificar la estrategia de búsqueda ya que garantiza un correcto punto de inicio para la identificación de los artículos a analizar, y su resultado.

En la realización de una primera búsqueda se seleccionaron las fuentes de información Google Scholar, IEEE Digital Library, ACM, Scopus, Science Direct, Springer, Wiley y otros. Luego, en una segunda etapa se realizó un refinamiento de la búsqueda escogiendo documentos de la fuente de información Scopus, ya que es una base de datos bibliográfica completa, cubriendo multiples áreas como ciencias, tecnología, medicina, ciencias sociales, entre otras, y además, incorpora bases de datos de otras fuentes de información como por ejemplo Google Scholar y/o Wiley. De igual manera, se definió de una mejor forma las palabras claves y la combinación de estas, acotando los artículos finales a analizar utilizando los protocolos de inclusión. En ambas etapas se utilizaron palabras claves que se detallan a continuación:

- **Criterio de inclusión 1**: El artículo debe presentar una estructura de datos compacta para la indexación de puntos.
- **Criterio de inclusión 2**: Las estructura de datos que se presentan en el artículo debe responder a una o más de las siguientes consultas: consulta de rango ortogonal entre las que están conteo en el rango y reporte si pertenece al rango, consulta de locación de puntos, y el

vecino más cercano o los vecinos más cercanos.

Luego de efectuar la selección y análisis de los artículos, se realizó una comparación teórica de las estructura de datos y una comparación empírica, mediante la experimentación de una estructura seleccionada.

3.1. Búsqueda con palabras claves

En una primera etapa de utilizaron las siguientes palabras claves: “Compact data structures”, “Succinct data structure”, Space efficient data structure”, “Multidimensional point set”, “Point set”, “Ortogonal range query” y “Nearest neighbor” donde estas se combinaron mediante conectores lógicos (AND y OR), los resultados de esta búsqueda arrojaron un total de 14.595 artículos.

Debido a la gran cantidad de artículos arrojados se acotó la búsqueda mediante los protocolos de inclusión, y la modificación de la consulta, la que finalmente se utilizó fue **”(TITLE-ABS-KEY ((”space efficient” OR succinct OR compact OR compressed OR implicit) W/3 ’data Structures”)) AND ((points OR point)) AND (nearest AND neighbors)”**, cabe mencionar que en la consulta sólo se incluyó la consulta de vecino más cercano (nearest neighbors) ya que es la consulta más compleja y la que resuelve una mayor cantidad de problemas, por lo que sólo se analizarán estas estructuras. De igual manera es importante registrar las estructuras de datos que respondan a las otras dos tipos de consultas ya que serán de utilidad para trabajos futuros. En esta segunda etapa el número final de artículos fue de 20, donde de estos se analizaron once de ellos seleccionados mediante los criterios de inclusión, como se muestra en 3.1. La búsqueda de los documentos fue realizada el día 02 de octubre del año 2018.

Tabla 3.1: Resultado de la búsqueda.

N°	Título artículo	Incluido	Excluido	Motivo Exclusión
1	Compressed Data Structures for Range Searching	✓		
2	Succinct geometric indexes supporting point location queries	✓		
3	Succinct quadtree for road data	✓		
4	A succinct, dynamic data structure for proximity queries on point sets	✓		
5	Approximate Nearest Neighbor Search in Metrics		x	Presenta estructura de datos no compacta y además en dos dimensiones
6	Compressed kd-tree for temporal graphs	✓		
7	Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon	✓		
8	Resolving SINR queries in a dynamic setting	✓		
9	Succinct and implicit data structures for computational geometry	✓		
10	Trajbase: Searching trajectories in multi-region	✓		

(Continúa en la siguiente página)

Tabla 3.1: Resultado de la búsqueda. (continuación)

N°	Título artículo	Incluido	Excluido	Motivo Exclusión
11	Windows into geometric events: Data structures for time-windowed querying of temporal point sets		x	No cumple con criterios de inclusión
12	Algorithm engineering: Concepts and practice		x	No presenta una estructura de datos compacta para la indexación de puntos.
13	Sparsely precomputing the light transport matrix for real-time rendering		x	No presenta una estructura de datos compacta para la indexación de puntos
14	Succinct geometric indexes supporting point location queries	Extensión del artículo número dos, por lo cual se analizará el más completo, es por esto el mismo título		
15	Sizing sketches: A rank-based analysis for similarity search		x	No presenta una estructura de datos compacta para la indexación de puntos
16	A compression scheme for the R-tree data structure	No se analizó, ya que no se pudo obtener su versión digital		
17	Succinct data structures for approximating convex functions with applications		x	No cumple con el criterio de inclusión número 2
18	Efficient density clustering method for spatial data		x	Si cumple con el criterio de inclusión número 1, pero no cumple con el criterio de inclusión número 2

(Continúa en la siguiente página)

Tabla 3.1: Resultado de la búsqueda. (continuación)

N°	Título artículo	Incluido	Excluido	Motivo Exclusión
19	Efficient search for approximate nearest neighbor in high dimensional spaces	✓		
20	Neighbours on a Grid	✓		

Cabe mencionar, que al momento de realizar la búsqueda entregó como resultado 20 artículos, de los cuales dos tienen el mismo título, esto es debido a que fueron presentados en diferentes eventos y uno es la extensión del otro, por lo que se analizó el documento más completo.

Capítulo 4

Estado del arte

He (2013) realizó trabajo similiar, donde analizó artículos publicados hasta el año 2013. En el presente trabajo se analizó su trabajo estudiando las estructuras de datos presentadas por él, lo que corresponde al 38 % de las analizadas en el presente documento.

En el trabajo de Caro et al. (2016), se propuso la estructura de datos llamada *compressed* k^d-tree ($c\mathsf{k}^\mathsf{d}$-tree) donde se abordó el almacenamiento y procesamiento de grafos web, mostrando el espacio utilizado por las estructuras de datos $c\mathsf{k}^\mathsf{d}$-tree, $bc\mathsf{k}^\mathsf{d}$-tree y k^d-tree con distintos conjuntos de datos (Dataset) en la Tabla 4.1. Los Dataset utilizados fueron Comm.Net que es un conjunto de datos sintéticos que simula comunicaciones cortas entre vértices aleatorios, Powerlaw también es sintético donde simula un grafo de grado de ley de potencia, donde pocos vértices tienen muchas más conexiones que otros vértices, Flickr-Day y Flickr-Sec son conjunto de datos con grafos temporales incrementales, que codifican el momento en que las personas se hicieron amigas en la red social de Flickr diferenciandose en la granularidad ya sea en días y segundos respectivamente, el conjunto de datos de Wiki-Links se compone de la historia de los enlaces entre los artículos de la versión en inglés de Wikipedia. Wiki-Edit es un grafo temporal, que indica el momento en que un usuario edita un artículo de Wikipedia, Yahoo-Netflow contiene registros de comunicación entre usuarios finales en Internet y servidores Yahoo! y para finalizar, el conjunto de datos de la Yahoo-Session es un grafo temporal que contiene el tiempo cuando un usuario busca un conjunto de términos de consulta en el buscador Yahoo! (Caro et al., 2016).

Las estructuras de datos utilizadas en el trabajo de Caro et al. se encuentran en el siguiente link: `https://github.com/diegocaro/cpp-btree`, posteriormente se mostrará el tiempo teórico utilizado por esta estructura para realizar ciertas consultas.

Tabla 4.1: Espacio utilizado por $c\mathsf{k}^\mathsf{d}$-tree, $bc\mathsf{k}^\mathsf{d}$-tree y k^d-tree, almacenando grafos web. Espacio medido en bits por contacto (bits total/contactos del grafo). Extraido de Caro et al. (2016)

Dataset	$c\mathsf{k}^\mathsf{d}$**-tree**	$bc\mathsf{k}^\mathsf{d}$**-tree**	k^d**-tree**
I-Comm.net	26.0	26.4	38.4
I-Powerlaw	31.9	32.6	48.8
I-Wiki.Links	61.2	62.2	92.0
I-Yahoo-Netflow	34.0	36.1	47.4
G-Flickr-Secs	46.1	46.8	71.0
G-Flickr-Days	23.0	24.3	28.6
P-Wiki-Edit	41.7	41.9	56.5
P-Yahoo-Session	47.1	45.2	46.0

4.1. Búsqueda de rango

El problema de búsqueda de rango ha sido estudiado de forma prolongada.

He (2013) realizó una investigación donde presenta distintos trabajos sobre estructuras de datos compactas a la hora de responder a ciertas consultas. Se presenta una representación de Munro (1979) de un $kd-tree$ implícito que responde a la consulta por rango ortogonal en tiempo $\mathcal{O}(n^{1-\frac{1}{d}+k})$, donde k es el número de puntos reportados, y en su construcción tarda $\mathcal{O}(n \cdot \log n)$. En un espacio bidimensional, Arroyuelo et al. (2011), proponen una estructura de datos implícita que responde la consulta de búsqueda de rango ortogonal en $\mathcal{O}(m \cdot \log n + k)$, donde puede ser construida en tiempo $\mathcal{O}(n^3)$, donde k es el número de puntos reportados y $m = \mathcal{O}(\sqrt{n})$ que es el

número mínimo de cadenas que se puede descomponer el conjunto. En relación a la consulta conteo en rango ortogonal en espacio bidimensional la estructura propuesta por Chazelle (1988) admite la consulta en $\mathcal{O}(\log n)$ solo si las coordenadas son enteras, la estructura de datos de JaJa et al. (2004) en tiempo $\mathcal{O}(\frac{\log n}{\log(\log n)})$ y la estructura de datos presentada por Bose et al. (2009) y Yu et al. (2011) utiliza $n \cdot \log n + o(n \cdot \log n)$ bits, responde a la consulta de reporte en rango ortogonal y conteo en rango ortogonal en tiempo $\mathcal{O}((k+1) \cdot \frac{\log n}{\log(\log n)})$ y $\mathcal{O}(\frac{\log n}{\log(\log n)})$ respectivamente, donde k es el número de puntos reportados y para su construcción logra el tiempo $\mathcal{O}(n \cdot \log n)$.

La estructura de datos dinámica en espacio d-dimensional de Matousek (1992) presentada en el documento de Aronov et al. (2018) se construye en espacio lineal y tiempo $\mathcal{O}(n \cdot \log n)$ admite consultas de reporte en rango ortogonal $\mathbb{R}^d$ en tiempo $\mathcal{O}(n^{1-\frac{1}{d}} \operatorname{polylog} n)$ y se puede eliminar e insertar en tiempo $\mathcal{O}(\log n)$ y $\mathcal{O}(\log^2 n)$ respectivamente.

Caro et al. (2016) proponen una estructura de datos llamada *compressed* $\mathsf{k^d}$-tree ($c\mathsf{k^d}$-tree) donde el espacio total ocupado es $m \cdot \log \frac{n^d}{m} + \mathcal{O}(k^d \cdot m)$ bits y responde a la consulta en tiempo $\mathcal{O}(m^{\frac{3}{4}})$, donde m es la cantidad de bits en uno almacenados en la estructura de datos.

4.2. Ubicación de un punto

La ubicación o locación de un punto fue la consulta de menor frecuencia en los artículos analizados con un total de 2.

Braun and Esswein (2014) en su trabajo, propone una estructura de datos compacta que puede crear m índices geométricos usando $o(n)$ bits, ésta responde la ubicación de puntos en triangulación planar en $\mathcal{O}(\log n)$. Además soporta la locación de puntos usando $\log(n) + 2 \cdot \sqrt{\log(n)} + \mathcal{O}(\log^{\frac{1}{4}} \cdot n)$ comparaciones de puntos con lineas y también admite la consulta en tiempo $o(\log n)$ cuando las coordenadas son números enteros.

Nosseir et al. (2012) propone mantener una lista de puntos codificados y además un índice espacial como por ejemplo un quad-tree en dos dimensiones. Los beneficios que trae esto es cuando la cantidad de números de trayectorias generadas por los puntos almacenados es alto, el número codificado permite realizarlo con un costo menor y acelera el procesamiento de consultas realizando

intersección de regiones.

4.3. Búsqueda de vecinos más cercanos

Dado un conjunto llamado N, de n puntos en el plano, la consulta del vecino más cercano en dos dimensiones devuelve el punto más cercano (en términos de distancia euclidiana la cual es $d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) a un punto de consulta dado (He, 2013).

Obtener el vecino más cercano es la consulta más estudiada en la actualidad ya que puede ser utilizadas de muchas maneras, es por eso que en esta revisión 5 de los 11 artículos abordaron esta consulta.He (2013) realizó un estudio, en el cuál analizó estructuras de datos compactas que respondan a las consultas abordadas en el presente trabajo. He obtuvo que Brönnimann et al. (2004) presenta una estructura de datos implícita que responde al vecino más cercano en dos dimensiones en un tiempo de $\mathcal{O}(\log^2 n)$, en tres dimensiones responde a la consulta de disparo "shooting" y "linear programming" en la intersección de semi-programas en tiempo $\mathcal{O}(\log^2 n)$ y la estructura de datos de Dobkin and Kirkpatrick (1990) responde a las mismas consultas en tiempo $\mathcal{O}(\log n)$. He (2013) aborda también el trabajo de Chan and Chen (2007) que presenta una estructura de datos eficiente en espacio que responde a la consulta del vecino más cercano en dos dimensiones en tiempo $\mathcal{O}(\log^{1,71} n)$.

Una consulta dentro de la familia de consulta de proximidad es el segmento más cercano que consiste en dado un punto encontrar no el punto más cercano, sino, el segmento más cercano a este. En relación a esta consulta Ishiyama et al. (2017) proponen una estructura de datos llamada quad-tree sucinto en dos dimensiones, donde esta responde los k puntos más cercanos en tiempo $T_a(Q, q, e+L)+\mathcal{O}(N(p, q, e+2L))+\mathcal{O}(k \cdot \log k)$, donde $N(p, q, e)$ es el número de puntos en p dentro de la distancia e desde q, $T_a(p, q, e)$ es el tiempo para enumerar los puntos y Q es un conjunto de puntos base en P y el segmento más cercano en tiempo $T_a(Q, q, e + M + L) + \mathcal{O}(N(p, q, e + M + 2 \cdot L)) + \mathcal{O}(k \cdot \log k)$, donde P es el conjunto de puntos en una cadena poligonal, Q es un conjunto base para puntos en P y M es el máximo largo del segmento de línea.

Venkat and Mount (2014) proponen un quad-tree sucinto en d dimensiones que ocupa $(d+2) \cdot m + o(m)$ bits, donde m es la cantidad total de nodos. Esta estructura de datos responde la

consulta del vecino más cercano aproximado en tiempo $\mathcal{O}((\frac{1}{\epsilon})^d \cdot (\log \Phi) \cdot \log^2(\log m))$, donde P es el conjunto de puntos en $\mathbb{R}^d$, ϵ es un parámetro de error mayor a 0 y Φ es el radio de aspecto de P; y la consulta de k-vecinos más cercanos aproximados en $\mathcal{O}(((\frac{1}{\epsilon})^d + k) \cdot (\log \Phi) \cdot \log^2(\log m))$, donde $k > 1$. Además, presentaron una versión dinámica de la estructura, donde para insertar y eliminar puntos lo realiza en tiempo amortizado $\mathcal{O}(log\Phi + b \cdot (\frac{\log N}{\log \log N}) + \log m)$ y $\mathcal{O}((b \cdot (\frac{\log N}{\log \log N}) + \log m) \cdot \log \Phi)$ respectivamente, al ésta transformarse en una estructura dinámica las consultas anteriores se deben multiplicar por un factor $\mathcal{O}(\frac{\log N}{\log \log N})$, donde $N = d \cdot b \cdot n$, d son la cantidad de dimensiones, n la cantidad de elementos y b es el largo de la cadena de bits de cada coordenada.

Agarwal et al. (2018) presentan una estructura de datos eficiente en espacio bidimensional y dinámica que admite consultas geodésicas del vecino más cercano para un conjunto de puntos S en un polígono P, la estructura presentada soporta la consulta en tiempo $\mathcal{O}(\log^2 n \cdot \log^2 m)$, las inserciones y eliminaciones toman un tiempo amortizado esperado de $\mathcal{O}(\log^5 n \cdot \log m + \log^4 n \cdot \log^3 m)$ y $\mathcal{O}(\log^7 n \cdot \log m + \log^6 n \cdot \log^3 m)$ respectivamente, el espacio utilizado por la estructura es $\mathcal{O}(n \cdot \log^3 n \cdot \log m + m)$,de igual forma, presentan una estructura de datos mejorada para cuando no existan eliminaciones, en esta configuración las consultas toman en el peor de los casos, tiempo $\mathcal{O}(\log^2 n \cdot \log^2 m)$ y las inserciones toman un tiempo amortizado $\mathcal{O}(\log n \cdot \log^3 m)$, donde espacio ocupado por esta versión es $\mathcal{O}(n \cdot \log n \cdot \log m + m)$, donde n es el número de sitios en S y m es el número de vértices en el polígono P.

Brodnik and Munro (1996) presentan una estructura de datos sucinta que responde el vecino más cercano en espacio bidimensional y n-dimensional, en dos dimensiones. Si el universo es una cuadrícula de MxM y N es un subconjunto, la realiza bajo la norma L_∞ en tiempo constante utilizando $M^2 + \frac{M^2}{2 \cdot \log n + \mathcal{O}(M^\epsilon)}$ bits, donde la norma L_∞ es $\underset{1 \leq i \leq n}{máx} |x_i|$ y x es un vector de puntos. Si el tamaño del universo máximo es M x M puntos y N es un subconjunto responde la consulta del vecino más cercano bajo la norma L_1 en tiempo constante utilizando $M^2 + \frac{M^2}{2 \cdot m} + M \cdot \frac{2 \cdot m + 1}{\sqrt{m}} + \mathcal{O}(M^\epsilon)$ donde la norma L_1 es $\sum_{i=1}^{\infty} |x_i|$, x es un vector de puntos, ϵ es cualquier número mayor a 0 y menor o igual a 1 y $M = 2^m$ donde m es el tamaño de una palabra. En un universo d-dimensional con M^d puntos y N un subconjunto responde a la consulta bajo la norma L_∞ en tiempo constante utilizando $M^d + \frac{M^d}{d \cdot \log M} + \mathcal{O}(M^\epsilon)$ bits de memoria, donde la norma L_∞ es $\underset{1 \leq i \leq n}{máx} |x_i|$ y x es un vector de puntos.

En el documento de Aronov et al. (2018) presenta la estructura de datos dinámica de

Kaplan et al. (2017) en espacio bidimensional que soporta la consulta del vecino más cercano en tiempo $\mathcal{O}(\log^2 n)$ en el peor caso, la inserción y eliminación toma tiempo amortizado $\mathcal{O}(\log^3 n)$ y $\mathcal{O}(\log^5 n)$ respectivamente.

4.4. Aplicaciones

De la totalidad de los documentos analizados algunos presentan análisis de los algoritmo de construcción y de consulta de forma teórica (Bille et al. (2015); Bose et al. (2008); Venkat and Mount (2014)) y otros se centran en análisis utilizando la estructura de datos en una aplicación práctica y/o real, como por ejemplo la representación de carreteras de una ciudad de Japón (Braun and Esswein (2014); Ishiyama et al. (2017)), representación de relaciones en redes sociales mediante grafos temporales, representación de puntos en polígonos respondiendo a consultas de distancia, (Agarwal et al. (2018)), representación de ubicación de transmisores de señal (Aronov et al. (2018)).

4.5. Discusión

La revisión aquí presentada aporta a la comunidad con un estudio actual, es decir, hasta el año 2018, incluyendo estructuras de datos propuestas no analizadas por He correspondiente al 62 % del total de estructuras de datos analizadas en el presente documento.

Luego de efectuar la revisión de los artículos se obtuvieron distintas conclusiones. Estructuras de datos compactas es un tema muy activo debido al aumento del big-data. Dentro de este tema el lugar donde más se investiga es Estados Unidos y Canadá con cuatro y tres artículos respectivamente de un total de once como se muestra en Figura 4.1, y de estos documentos la mayoría a desarrollado y/o presentado estructura de datos compactas en dos dimensiones como se muestra en Figura 4.2, de esto se deduce que no hay una gran cantidad de trabajos enfocados en la multidimensionalidad. La mayoría de las estructuras de datos compactas propuestas son estáticas, quiere decir que luego de crearla no se pueden insertar ni eliminar datos como se muestra en la Figura 4.2.

Debido a la consulta principal enfocada en este trabajo que es la de proximidad se realizó un primer resumen sólo abarcando esta consulta ilustrado en la Tabla 4.2. En dicha fase se indica

la consulta que resuelve ya sea el vecino más cercano (NN), los k-vecino más cercano (k-NN) y el segmento más cercano (NS), los autores de las estructuras de datos, el tiempo de consulta, la memoria ocupada (donde N/I significa no informada por los autores), las dimensiones, donde 2d significa espacio bidimensional y d significa más de dos dimensiones) y para finalizar se representa si la estructura es dinámica o estática indicando D y E respectivamente. Llama la atención que la estructura de Brodnik and Munro (1996), responde en tiempo constante la consulta, diferenciándose de las otras estructuras por el espacio ocupado para almacenarla, llegando a ser la que más espacio necesita.

En la Tabla 4.3, se muestra un resumen de las estructuras de datos que responden a las consultas restantes que son búsqueda de rango ortogonal (OSR), conteo de rango ortogonal (OCR) y locación de un punto (LP), donde en ésta se incluye las mismas columnas presentadas en la tabla 4.2, la importancia del registro de estas consultas es para la realización de trabajos futuros.

Tabla 4.2: Resumen de la búsqueda para la consulta de proximidad.

	Autor(es)	**Tiempo consulta**	**Memoria ocupada**	**Dim.**	**Estática / Dinámica**
NN	Brönnimann et al. (2004)	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n)$ bits	2	E
	Chan and Chen (2007)	$\mathcal{O}(\log^{1,71} n)$	$\mathcal{O}(n)$ bits	2	E
	* Venkat and Mount (2014)	$\mathcal{O}((\frac{1}{\epsilon})^d \cdot (\log \Phi) \cdot \log^2(\log m))$	$(d+2) \cdot m + o(m)$ bits, m cantidad de nodos m=n	d	E
	* Venkat and Mount (2014)	$\mathcal{O}((\frac{1}{\epsilon})^d \cdot (\log \Phi) \cdot \log^2(\log m)) \cdot \mathcal{O}(\frac{\log n}{\log \log n})$	$(d+2) \cdot m + o(m)$ bits, m cantidad de nodos m=n	d	D

(Continúa en la siguiente página).

*estructura de datos no analizada por He (2013)

Tabla 4.2: Resumen de la búsqueda. (continuación).

	Autor(es)	**Tiempo consulta**	**Memoria ocupada**	**Dim.**	**Estática / Dinámica**
	* Agarwal et al. (2018)	$\mathcal{O}(\log^2 n \cdot \log^2 m)$	$\mathcal{O}(n \cdot \log^3 n \cdot \log m + m)$ bits	2	E
	* Brodnik and Munro (1996)	Tiempo cte.	Norma L_∞ : $M^2 + \frac{M^2}{2 \cdot \log n + \mathcal{O}(M^\epsilon)}$ bits. Norma $L_1 : M^2 + \frac{M^2}{2} + M \cdot \frac{2 \cdot m + 1}{\sqrt{m}} + \mathcal{O}(M^\epsilon)$ bits. $\epsilon > 0$,$y \leq 1$, $M = 2^m$ y m =tamaño de una palabra	2	E
	* Brodnik and Munro (1996)	Tiempo cte.	Norma L_∞ : $M^2 + \frac{M^2}{2 \cdot \log n + \mathcal{O}(M^\epsilon)}$ bits. Norma $L_1 : M^2 + \frac{M^2}{2} + M \cdot \frac{2 \cdot m + 1}{\sqrt{m}} + \mathcal{O}(M^\epsilon)$ bits. $\epsilon > 0$,$y \leq 1$, $M = 2^m$ y m =tamaño de una palabra	d	E
	* Kaplan et al. (2017)	$\mathcal{O}(\log^2 n)$	N/I	2	D

(Continúa en la siguiente página).

*estructura de datos no analizada por He (2013)

Tabla 4.2: Resumen de la búsqueda. (continuación).

	Autor(es)	**Tiempo consulta**	**Memoria ocupada**	**Dim.**	**Estática / Dinámica**
k-NN	* Ishiyama et al. (2017)	$T_a(Q,q,e + L) + \mathcal{O}(N(p,q,e + 2L)) + \mathcal{O}(k \cdot \log n)$	$n \cdot w + \mathcal{O}(n \cdot \log n)$ bits, w=bits necesarios para representar un punto (≥ 64)	d	E
	* Venkat and Mount (2014)	$\mathcal{O}(((\frac{1}{\epsilon})^d + k) \cdot (\log \Phi) \cdot \log^2(\log m))$	$(d+2) \cdot m + o(m)$ bits, m cantidad de nodos m=n	d	E
	* Venkat and Mount (2014)	$\mathcal{O}(((\frac{1}{\epsilon})^d + k) \cdot (\log \Phi) \cdot \log^2(\log m)) \cdot \mathcal{O}(\frac{\log n}{\log \log n})$	$(d+2) \cdot m + o(m)$ bits, m cantidad de nodos m=n	d	D
NS	* Ishiyama et al. (2017)	$T_a(Q,q,e + M + L) + \mathcal{O}(N(p,q,e+M+2L)) + \mathcal{O}(k \cdot \log k)$	$n \cdot w + \mathcal{O}(n \cdot \log n)$ bits, w=bits necesarios para representar un punto (≥ 64)	d	E

*estructura de datos no analizada por He (2013)

Tabla 4.3: Resumen de la búsqueda para las consultas de búsqueda de rango y ubicación de puntos.

	Autor(es)	**Tiempo consulta**	**Memoria ocupada**	**Dim.**	**Estática / Dinámica**
OSR	Munro (1979)	$\mathcal{O}(n^{1-\frac{1}{d}}+k)$	N/I	d	E
	Arroyuelo et al. (2011)	$\mathcal{O}(m \cdot \log n + k)$, m=$\mathcal{O}(\sqrt{n})$	N/I	2	E
	Bose et al. (2009) y Yu et al. (2011)	$\mathcal{O}((k+1)\cdot\frac{\log n}{\log\log n})$	$n\cdot\log n + o(n\cdot\log n)$ bits	2	E
	* Matousek (1992)	$\mathcal{O}(n^{1-\frac{1}{d}}\cdot \text{polylog}\, n)$	N/I	d	E
	* Caro et al. (2016)	$\mathcal{O}(m^{\frac{3}{4}})$	$m\cdot\log\frac{n^d}{m} + \mathcal{O}(k^d\cdot m)$ bits	d	E
OCR	Chazelle (1988)	$\mathcal{O}(\log n)$, si coordenadas son números enteros	N/I	2	E
	JaJa et al. (2004)	$\mathcal{O}(\frac{\log n}{\log\log n})$	N/I	2	E
	Bose et al. (2009) y Yu et al. (2011)	$\mathcal{O}(\frac{\log n}{\log\log n})$	$n\cdot\log n + o(n\cdot\log n)$ bits	2	E
LP	* Bose et al. (2008)	$\mathcal{O}(logn)$ y $o(\log n)$, si coordenadas son enteros	$o(n)$ bits	2	E

El tiempo utilizado para responder a las consultas es de relevancia a la hora se definir que estructura de datos es mas conveniente utilizar, de las analizadas en el presente trabajo, la que menos tiempo tarda en responder (tiempo constante) a las consultas es la estructura de Brodnik and Munro (1996).

Si la limitación es espacio, es decir memoria, a la hora de escoger una estructura de datos

*estructura de datos no analizada por He (2013)

la que ocupa menos espacio ($\mathcal{O}(n)$ bits) es la de Brönnimann et al. (2004) y la de Chan and Chen (2007).

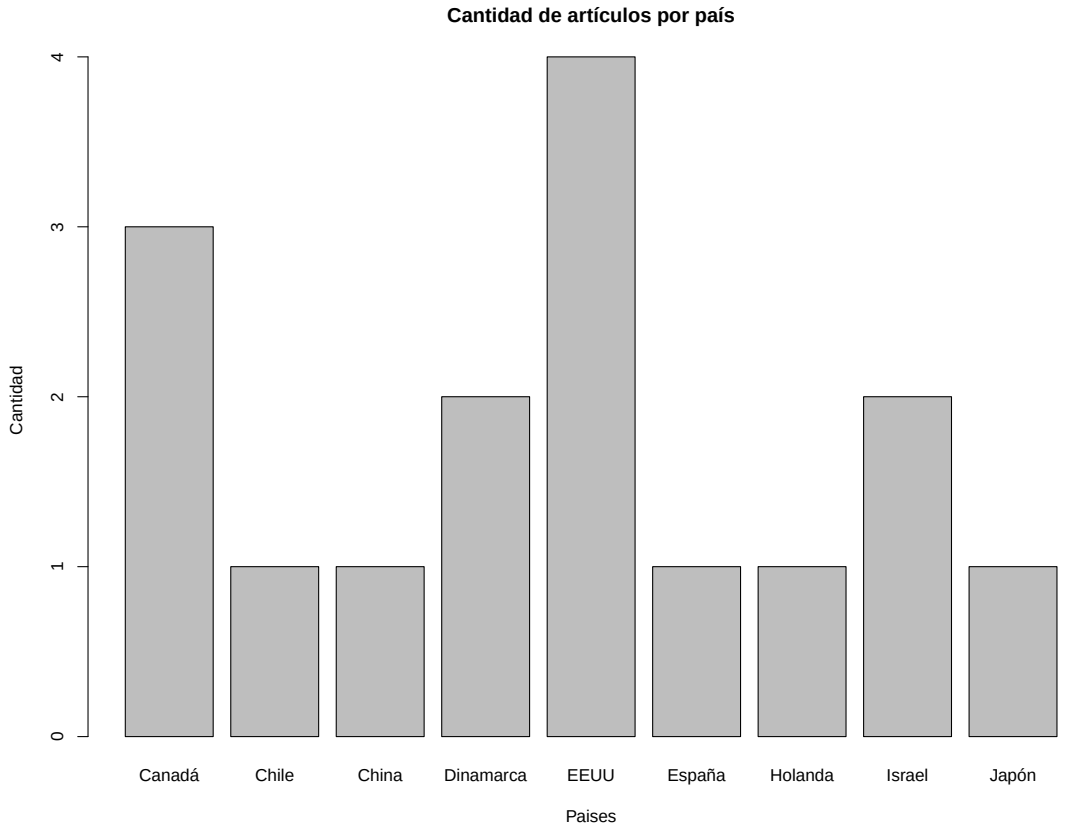

Figura 4.1: Cantidad de artículos por país

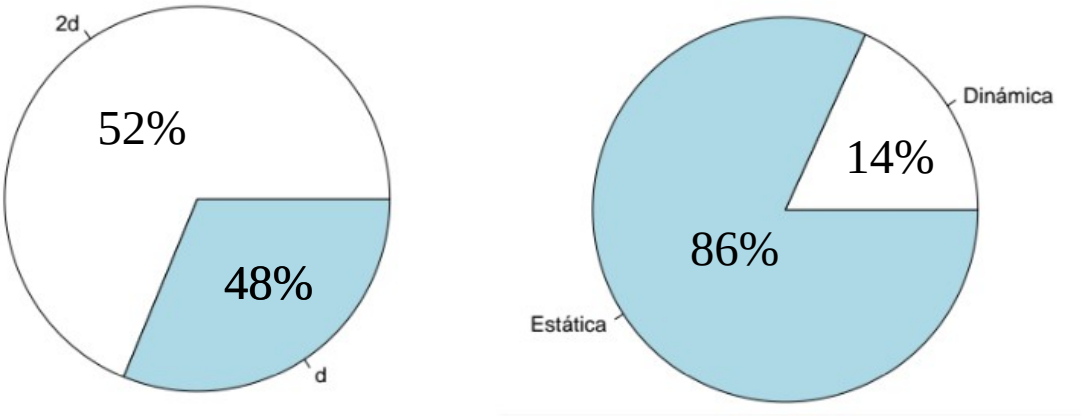

Figura 4.2: El gráfico de la izquierda representa la comparación de las estructura de datos presentadas en 2d v/s d-dimensiones y al lado derecho, la relación entre estructuras dinámicas v/s estáticas.

Capítulo 5

Experimentación

Luego de realizar el estudio de estructuras de datos para diferentes consultas, se realizó la etapa de experimentación donde esta buscaba implementar una estructura de datos para estudiar como se comporta en la práctica en relación a espacio y tiempo.

La estructura de datos compacta escogida para la experimentación fue la de Caro et al., debido a que fue estudiada en esta búsqueda y además, está disponible en el repositorio del autor, su implementación está realizada en el lenguaje de programación C++ 11 y disponible para d-dimensiones.

Además, la estructura de datos escogida cuenta con una posibilidad de aportar a la comunidad con trabajos futuros, como la implementación de una nueva consulta, ya que esta sólo soporta la consulta de rango.

Se utilizaron 12 colecciones sintéticas de puntos aleatorios entre 0 y 100.000 como se muestra en la Tabla 5.1, diferenciándose en la cantidad de puntos $(10^4, 10^5, 10^6)$, en relación a las dimensiones en una primera etapa se intentó probar en 3, 9 y 27 dimensiones, lo que no fue posible completar ya que la estructura de datos esta pensada hasta la dimensión 5.

Tabla 5.1: Colección de datos sintéticos aleatorios

Colección	**Cantidad de puntos (n)**	**Dimensiones**
1	10.000	1
2	10.000	2
3	10.000	3
4	10.000	4
5	100.000	1
6	100.000	2
7	100.000	3
8	100.000	4
9	1.000.000	1
10	1.000.000	2
11	1.000.000	3
12	1.000.000	4

En la Tabla 5.2 se muestra el tamaño de los conjuntos, el tamaño de salida de datos donde ya comprimió la estructura utilizando el índice $ck^d - tree$ y el porcentaje de compresión utilizando el ratio de compresión. En ciertas colecciones como la 12 el porcentaje de compresión es bajo $(3,56\,\%)$ e incluso en colecciones como la 4 y la 8 simbolizadas por (*) no comprime incluso es mayor el tamaño de salida es por eso el signo negativo en el porcentaje. Finalmente muestra la entropía total de los datos.

En la Figura 5.1 se ilustra el porcentaje de compresión de la estructura de datos a medida que aumenta la cantidad de puntos. Los cuatro colores simbolizan las dimensiones en la que están estos puntos, en morado en 1 dimensión, en amarillo 2 dimensiones, en celeste 3 dimensiones y finalmente en verde 4 dimensiones. Se concluye que la estructura posee una gran compresión en colecciones en bajas dimensiones ya sea en una o en dos, ya que debido al algoritmo de construcción de la estructura, en dimensiones mayores a 2 disminuye el porcentaje de compresión.

Tabla 5.2: Compresión de la estructura de datos

Dimensión	**Tamaño entrada de datos**	**Tamaño salida de datos**	**% de compresión**	**Entropía total**
1d	69,0 kB	**9,7 kB**	85,9 %	16,5 kB
2d	127,8 kB	51,7 kB	59,5 %s	**32,8 kB**
3d	186,7 kB	125,6 kB	32,7 %	**49,5 kB**
4d	245,5 kB	273,3 kB	−10,5 %(*)	**66 kB**
1d	689,0 kB	**24,8 kB**	96,4 %	197,2 kB
2d	1,3 MB	426,7 kB	96,7 %	**329,8 kB**
3d	1,9 MB	1,1 MB	42,1 %	**0,59 MB**
4d	2,5 MB	2,6 MB	−4 % (*)	**0,79 MB**
1d	6,9 MB	**26,7 kB**	99,9 %	2,1 MB
2d	12,8 MB	**2,4 MB**	81,2 %	4,04 MB
3d	18,7 MB	10,2 MB	45,4 %	**6,20 MB**
4d	24,6 MB	23,7 MB	3,65 %	**8,27 MB**

En la Tabla 5.3 se presenta los tiempos de CPU de ejecución medidos en microsegundos a la consulta de rango, en las distintas dimensiones soportadas por la estructura, para distintas cantidades de puntos (10.000, 100.000 y 1.000.000) y distintos rangos, que fueron escogidos de forma aleatoria aumentando el área de consulta, la elección de los rangos no influyen en la medición de tiempos ya que los puntos fueron generados al azar uniformemente.

Para que el resultado en tiempo sea lo más exacto posible se realizó la misma consulta sobre la misma colección, utilizando 50 distintos rangos de consulta, promediando el tiempo acumulado.

La presente experimentación se realizó en una máquina con las siguientes características: memoria 7,7 GiB, procesador Intel Core i7-5500U CPU 2.40GHz x4, sistema operativo Linux y distribución UBUNTU 18.04.

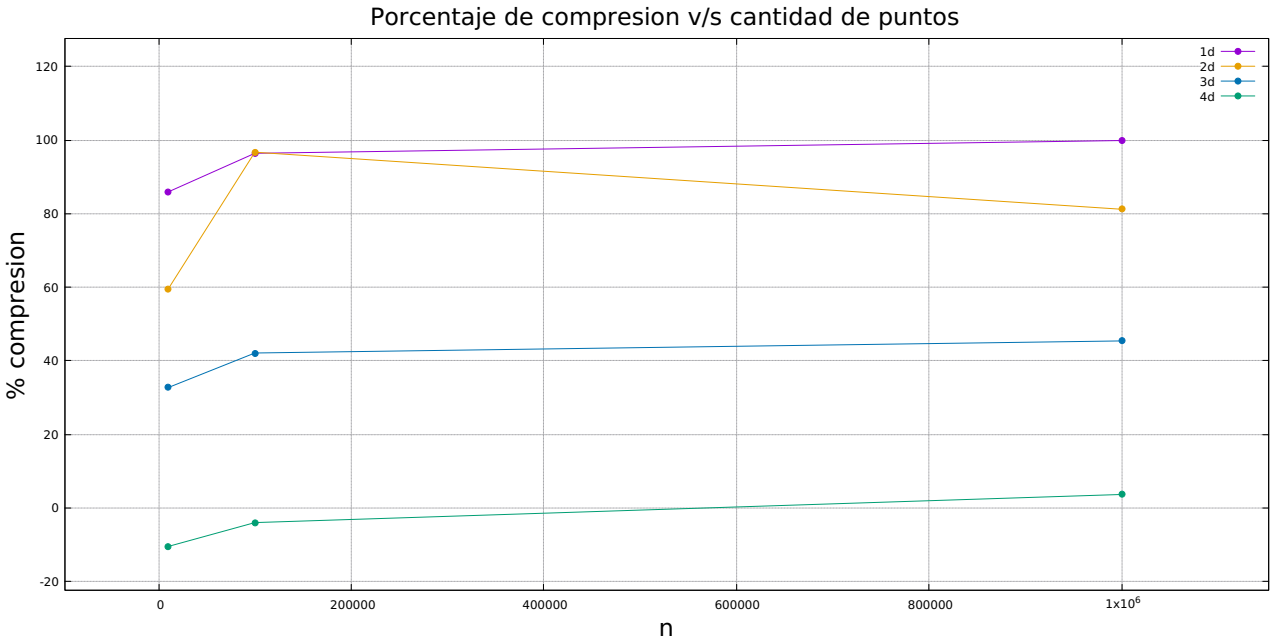

Figura 5.1: Porcentaje de compresión v/s cantidad de puntos.

Tabla 5.3: Comparación del tiempo promedio en microsegundos en realizar la consulta por rango para distintos tamaños de rango de consulta, distintas dimensiones y distinta cantidad de puntos sintéticos aleatorios

Tipo Consulta		**Rango**		
Dimensión	**Área**	$n = 10^4$	$n = 10^5$	$n = 10^6$
1	10^1	8	7	7
	10^2	15	32	36
	10^3	116	284	323
2	10^1	4	5	6
	10^2	5	6	12
	10^3	13	58	380
3	10^1	6	7	8
	10^2	6	8	9
	10^3	8	13	33

	10^1	10	12	14
4	10^2	10	12	15
	10^3	12	15	21

Las dimensiones en la que están los datos es de gran relevancia a la hora de analizar el tiempo ocupado, ya que con una dimensión alta se tiene un espacio mayor en el cuál se debe realizar la consulta, en el presente experimento se concluyó que la estructura de datos estudiada responde en menor tiempo en dimensiones altas y con una cantidad de puntos (n) menor, debido que los puntos están más dispersos. De la misma forma en las Figuras 5.2, 5.3, 5.4, 5.5, se muestra la comparación entre cantidad de puntos almacenada en la estructura de datos v/s milisegundos que tarda la consulta con determinados rangos (10^1, 10^2 y 10^3). Reafirmando la conclusión que en dimensiones altas tarda menos en responder a la consulta independiente sea el rango de consulta.

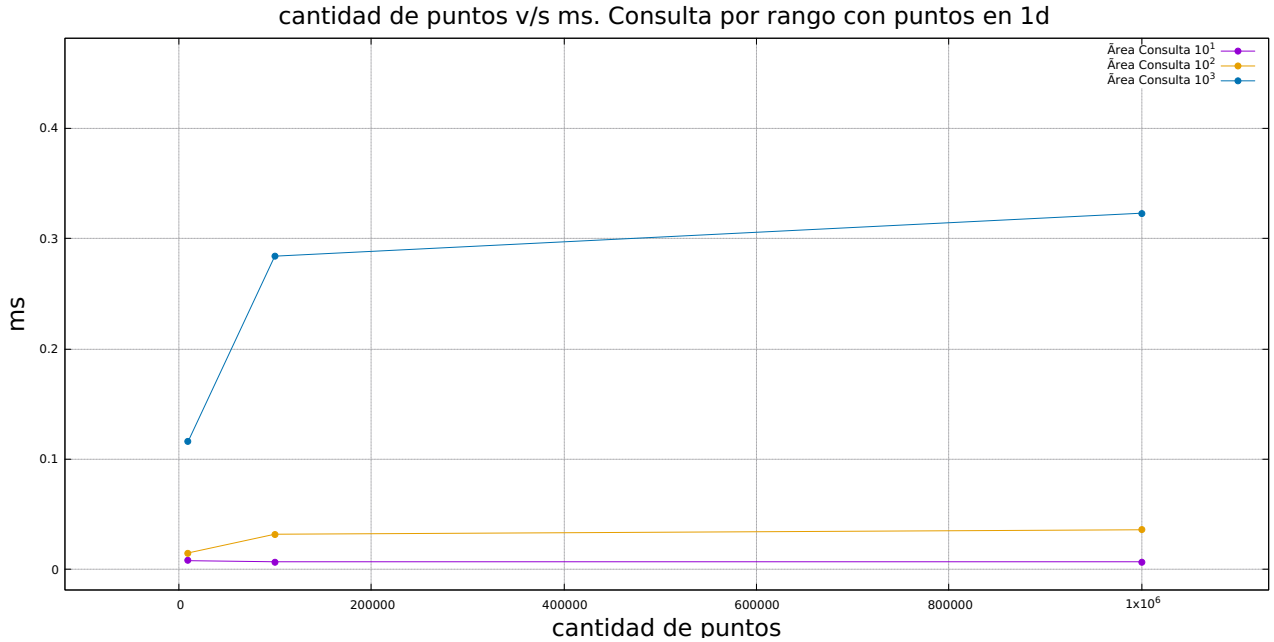

Figura 5.2: cantidad de puntos v/s milisegundos en consulta por rango en 1 dimensión.

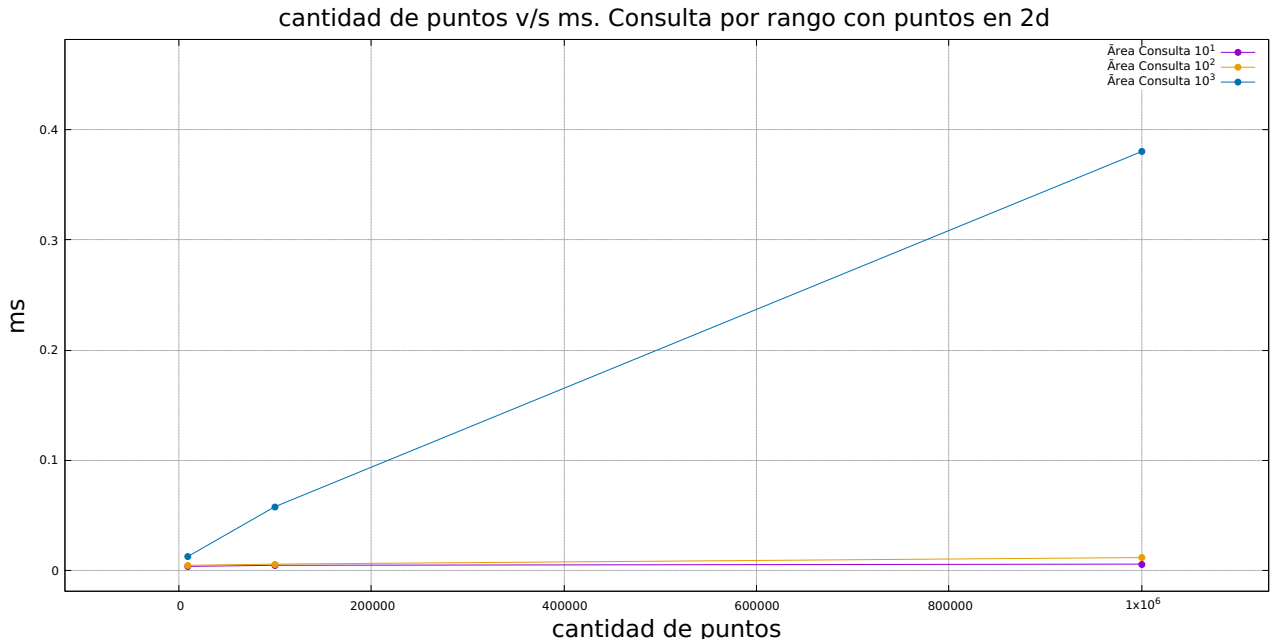

Figura 5.3: cantidad de puntos v/s milisegundos en consulta por rango en 2 dimensiones.

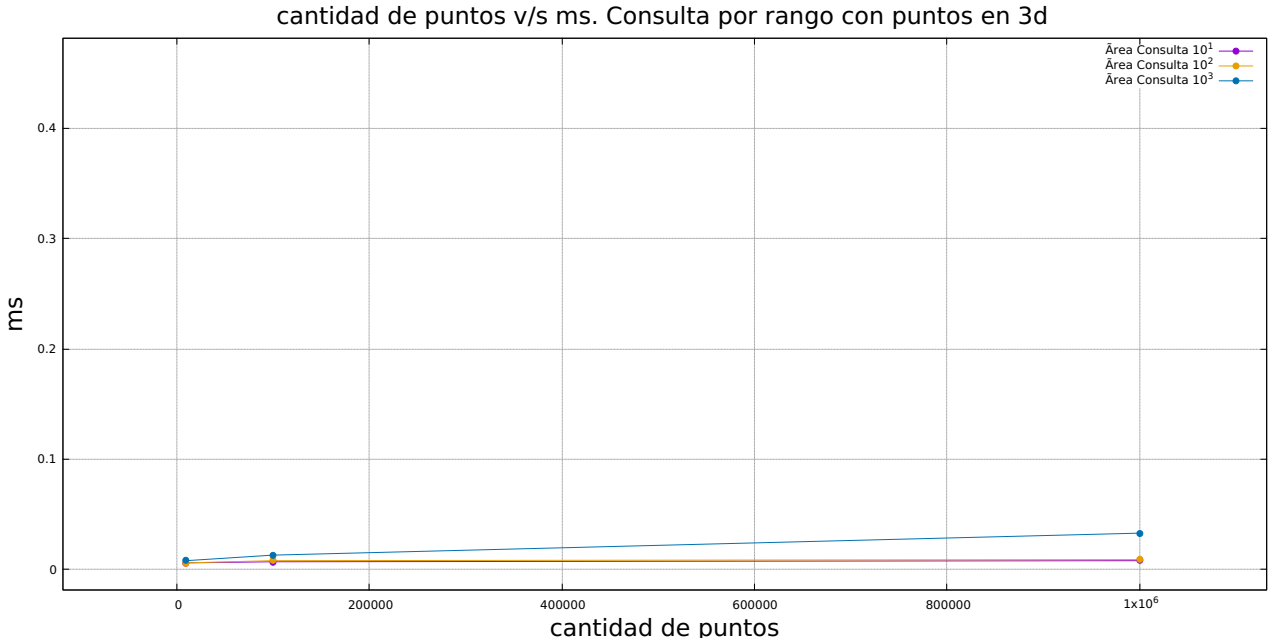

Figura 5.4: cantidad de puntos v/s milisegundos en consulta por rango en 3 dimensiones.

Figura 5.5: cantidad de puntos v/s milisegundos en consulta por rango en 4 dimensiones.

Capítulo 6

Conclusiones y trabajos futuros

La indexación de puntos ha sido un área de investigación y desarrollo activa durante los últimos tiempos. Actualmente, dado la necesidad de almacenar grandes cantidades de información a motivado un aumento en la investigación de estructuras de datos compactas, y por consiguiente, en la necesidad de contar con nuevos y más eficientes algoritmos de consulta sobre estas estructuras de datos.

Una primera aportación de este trabajo fue la realización de experimentación de la estructura de Caro et al. con respecto a puntos multidimensionales ya que sólo la realizan sobre grafos. Está experimentación produjo resultados en relación a tiempo de respuesta sobre consulta de rango y el tamaño de la estructura de datos compacta comparándola con los puntos no compactos, obteniendo como conclusión que esta implementación no comprime en dimensiones grandes como por ejemplo 4 y 5 debido al algoritmo de construcción de este. Y en dimensiones bajas como por ejemplo 1 o 2 puede llegar a comprimir hasta un 99 %, incluso estando por debajo de la entropía total de los datos.

La principal contribución de este trabajo es la realización de una revisión de la literatura y actualización ya que solo había una realizada por He (2013) donde solo abarcaba hasta el año 2013, por lo que no era actual. En la presente revisión se analizó las estructuras expuestas en su trabajo además del estudio de documentos que el no incluyó.

6.1. Trabajos Futuros

La etapa de búsqueda de los artículos fue enfocada en la consulta de vecinos más cercanos, debido a que es la consulta que resuelve una mayor cantidad de problemas y más complejos, por lo que a modo de trabajo futuro se propone agrandar el rango de búsqueda incluyendo la consulta de rango ortogonal y de locación de puntos, de igual manera se registraron los documentos que presentaban estas dos últimas consultas.

Al momento de seleccionar una estructura de datos de los artículos analizados, se escogió la de Caro et al. (2016), donde esta sólo responde a la consulta de rango y hasta 5 dimensiones, por lo que se propone la implementación de la consulta de vecinos más cercanos y aumentar las dimensiones soportadas.

Finalmente se propone la implementación de una nueva estructura de datos compacta que mejore los rendimientos de las que fueron analizadas en el presente trabajo.

Referencias

Agarwal, P. K., Arge, L., and Staals, F. (2018). Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon. *CoRR*, abs/1803.0.

Aronov, B., Bar-On, G., and Katz, M. J. (2018). Resolving SINR queries in a dynamic setting. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 107.

Arroyuelo, D., Claude, F., Dorrigiv, R., Durocher, S., He, M., López-Ortiz, A., Munro, J. I., Nicholson, P. K., Salinger, A., and Skala, M. (2011). Untangled monotonic chains and adaptive range search. *Theoretical Computer Science*, 412(32):4200–4211.

Bille, P., Gørtz, I. L., and Vind, S. (2015). Compressed Data Structures for Range Searching. In Dediu, A.-H., Formenti, E., Martín-Vide, C., and Truthe, B., editors, *Language and Automata Theory and Applications*, pages 577–586, Cham. Springer International Publishing.

Bose, P., He, M., Maheshwari, A., and Morin, P. (2009). Succinct Orthogonal Range Search Structures on a Grid with Applications to Text Indexing. In Dehne, F., Gavrilova, M., Sack, J.-R., and D., T. C., editors, *Algorithms and Data Structures*, pages 98–109, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bose, P., Y. Chen, E., He, M., Maheshwari, A., and Morin, P. (2008). Succinct Geometric Indexes Supporting Point Location Queries. *ACM Transactions on Algorithms*, 8.

Braun, R. and Esswein, W. (2014). Extending BPMN for Modeling Resource Aspects in the Domain of Machine Tools. *WIT Transactions on Engineering Sciences*, 87:450–458.

Brodnik, A. and Munro, J. I. (1996). Neighbours on a grid. In Karlsson, R. and Lingas, A., editors, *Algorithm Theory — SWAT'96*, pages 309–320, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brönnimann, H., Chan, T. M., and Chen, E. Y. (2004). Towards In-place Geometric Algorithms and Data Structures. In *Proceedings of the Twentieth Annual Symposium on Computational*

Geometry, SCG '04, pages 239–246, New York, NY, USA. ACM.

Caro, D., Rodríguez, M. A., Brisaboa, N. R., and Fariña, A. (2016). Compressed k d-tree for temporal graphs. *Knowledge and Information Systems*, 49(2):553–595.

Chan, T. M. and Chen, E. Y. (2007). In-Place 2-d Nearest Neighbor Search.

Chazelle, B. (1988). Functional Approach to Data Structures and Its Use in Multidimensional Searching. *SIAM J. Comput.*, 17(3):427–462.

Dobkin, D. P. and Kirkpatrick, D. G. (1990). Determining the separation of preprocessed polyhedra — A unified approach. In *Automata, Languages and Programming - l7th International Colloquium, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 400–413, Germany. Springer Verlag.

E. Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423.

He, M. (2013). Succinct and Implicit Data Structures for Computational Geometry. In Brodnik, A., López-Ortiz, A., Raman, V., and Viola, A., editors, *Space-Efficient Data Structures, Streams, and Algorithms: Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, pages 216–235. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ishiyama, K., Kobayashi, K., and Sadakane, K. (2017). Succinct Quadtrees for Road Data. In Beecks, C., Borutta, F., Kröger, P., and Seidl, T., editors, *Similarity Search and Applications*, pages 262–272, Cham. Springer International Publishing.

Jacobson, G. (1989). Space-efficient Static Trees and Graphs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 549–554, Washington, DC, USA. IEEE Computer Society.

JaJa, J., Mortensen, C. W., and Shi, Q. (2004). Space-Efficient and Fast Algorithms for Multidimensional Dominance Reporting and Counting. In *Proceedings of the 15th International Conference on Algorithms and Computation*, ISAAC'04, pages 558–568, Berlin, Heidelberg. Springer-Verlag.

Kaplan, H., Mulzer, W., Roditty, L., Seiferth, P., and Sharir, M. (2017). Dynamic Planar Voronoi Diagrams for General Distance Functions and Their Algorithmic Applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2495–2504, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Matousek, J. (1992). Efficient partition trees. *Discrete & Computational Geometry*, 8(1):315–334.

Munro, J. I. (1979). A multikey search problem. In *Proceedings of the 17th Allerton Conference on Communication, Control and Computing*, pages 257–276.

Navarro, G. (2016). *Compact Data Structures: A Practical Approach*. Cambridge University Press.

Nosseir, A., Flood, D., Harrison, R., and Ibrahim, O. (2012). Mobile Development Process Spiral.

Venkat, P. and Mount, D. M. (2014). A Succinct, Dynamic Data Structure for Proximity Queries on Point Sets. In *CCCG*.

Yu, C.-C., Hon, W.-K., and Wang, B.-F. (2011). Improved data structures for the orthogonal range successor problem. *Computational Geometry*, 44(3):148–159.