Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la Computación

REPRESENTACIÓN COMPACTA PARA SERIES DE TIEMPO DE RÁSTERS

Tesis para optar al grado de
MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

POR
Nataly Scarleth Cruces Saavedra
CONCEPCIÓN, CHILE
Noviembre, 2018

Profesor guía: Diego Seco Naveiras
Departamento de Ingeniería Informática y Ciencias de la Computación
Facultad de Ingeniería
Universidad de Concepción

Resumen

Los Sistemas de Información Geográfica (SIG) permiten almacenar y modelar conjuntos de datos espaciales en formato digital, los que permiten representar características de una determinada superficie en un instante de tiempo, como las precipitaciones, la temperatura, la densidad de población, entre otros. Los SIG almacenan sus datos en un modelo vectorial o en un modelo de ráster. Este último consiste en dividir el espacio en un conjunto regular de celdas, formando una matriz bidimensional con valores numéricos que representan alguna característica de la superficie en un determinado instante de tiempo, como por ejemplo, la temperatura de ese lugar en un determinado instante.

Los SIG permiten representar amplias extensiones de terreno generando enormes volúmenes de datos espaciales, lo cual supone un problema: el requerir mucho espacio de almacenamiento. Este problema es todavía más grave cuando se considera la variable temporal ya que se tiene, en el caso del modelo de ráster, un ráster por cada instante de tiempo.

Al tener una secuencia de rásters que representen una región en diferentes instantes de tiempo (lo que se conoce como serie de tiempo de rásters), se tiene la propiedad de que sus valores son similares entre sí, teniendo localidad temporal (además de la localidad espacial intrínseca en el modelo ráster). Es por esto que el objetivo de este trabajo de tesis es crear una representación compacta para series de tiempo de ráster, explotando la propiedad de localidad temporal en una serie de tiempo de rásters.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Capítulo 1

INTRODUCCIÓN

1.1 Motivación

Un Sistema de Información Geográfica (SIG) corresponde a una herramienta diseñada para realizar múltiples funciones, como almacenar, visualizar, consultar, analizar y modelar un conjunto de datos espaciales en formato digital, los que permiten representar características de una determinada superficie en un instante de tiempo, como las precipitaciones, la temperatura, la altitud; o variables sociales, como el número de habitantes, densidad de población, entre otros.

Cuando hacemos referencia a un sistema de información geográfica como administrador de datos espaciales también estamos tratando información temporal, aunque de manera implícita. La dimensión temporal adquiere un nuevo relieve en los estudios territoriales con las nuevas tecnologías de la información, los sistemas de observación de la Tierra y los sistemas de información geográfica.

En un SIG temporal también podemos utilizar un tiempo futuro en el que visualicemos situaciones espacio-temporales que pueden llegar a suceder. Podemos plantear este tipo de análisis en temas en los que recreemos situaciones futuras a partir de datos presentes o pasados.

El almacenamiento de los datos en un sistema de información geográfica puede ser basado en un modelo vectorial o en un modelo de ráster [1] (figura 1.1).

En un modelo vectorial los diferentes objetos (líneas, puntos o superficies) se relacionan mediante un modelo topológico de líneas, nodos y cadenas que determinan polígonos asociados a atributos, como se puede ver en la figura 1.2.

Por otro lado, el formato ráster se fundamenta en la división del área de estudio en una matriz de celdas. Cada una de estas celdas recibe un único valor que se considera representativo para toda la superficie abarcada por la misma (figura 1.3).

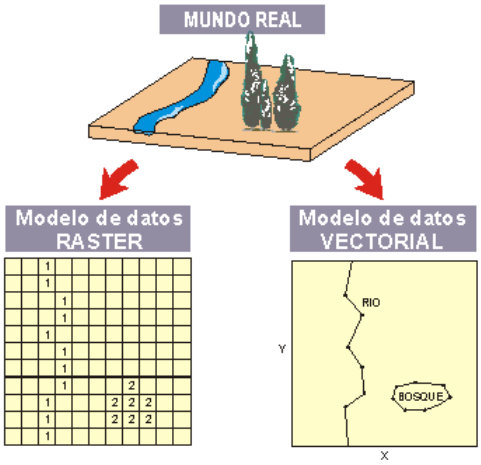

Figura 1.1: Almacenamiento en modelo ráster y modelo vectorial (Fuente [8]).

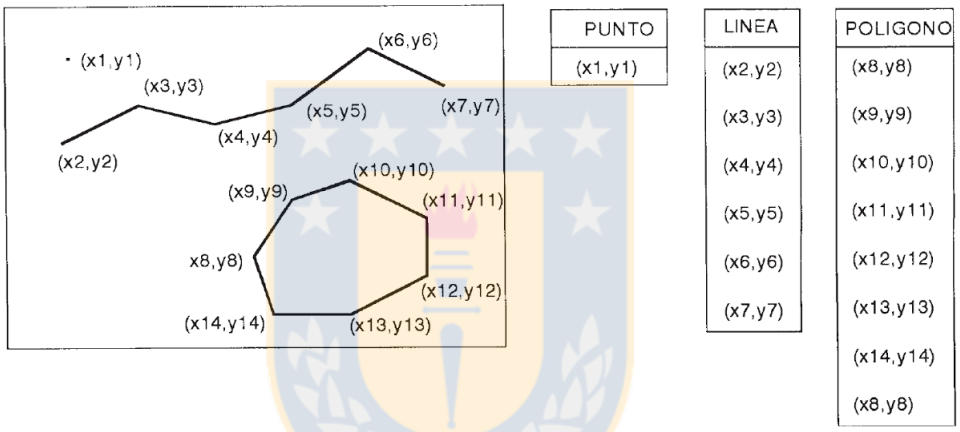

Figura 1.2: Modelo Vectorial: Representación de los elementos espaciales vectoriales (punto, línea y polígono) en un sistema de coordenadas cartesiano (Fuente [9]).

Cabe destacar que mientras más celdas representen una región, se tendrá una representación más detallada de ésta; pero a la vez implica que se tendrá un mayor volumen de datos que deberán ser almacenados. Cada matriz representa el estado de una región en un instante determinado, por lo que si se desea tener la evolución de los datos a través del tiempo, se necesita tener una sucesión de rásters, conocida como serie de tiempo de rásters.

Ejemplos de sistemas de información geográfica donde se utiliza el modelo de ráster, son para mapas de temperatura, de altitud, de superficie, de población, entre otros. La utilización de uno u otro modelo dependerá de los tipos de fuentes de datos

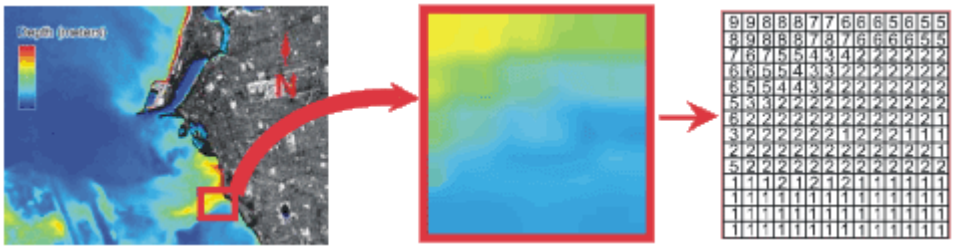

Figura 1.3: Modelo ráster (Fuente [10]).

y de la aplicación concreta que busquemos, pero en este trabajo nos enfocaremos en el modelo de ráster.

En los modelos de sistemas de información geográfica, podemos encontrar que los valores de los datos almacenados tienen localidad espacial, cumpliendo la primera ley de la geografía: "*Las cosas más próximas en el espacio, tienen una relación mayor que las distantes, aunque todas están relacionadas*" [2]. Esto quiere decir que, por ejemplo, en un ráster los valores que están cercanos entre sí, tienen valores similares. Además de la localidad espacial, podemos encontrar localidad temporal en una serie de tiempo de rásters, donde cada celda de un ráster tiene un valor similar a la misma celda en el ráster generado en el tiempo siguiente.

Los SIG permiten representar amplias extensiones de terreno con enormes volúmenes de datos espaciales, lo cual genera el problema de que se requiere mucho espacio para poder almacenarlo, más aún cuando se considera la variable temporal ya que se tiene, en el caso del modelo de ráster, un ráster por cada instante de tiempo. Es por esto que el objetivo de éste trabajo de tesis es crear una representación compacta para series de tiempo de rásters, explotando la propiedad de localidad temporal entre ellos, para poder así reducir el costo de almacenamiento de una secuencia de rásters. Además de reducir el costo de almacenamiento, se podrá realizar consultas espacio-temporales sin tener que descomprimir los datos. Hay varios tipos de consultas, pero nos centraremos en consultas de acceso, rango espacial y temporal. Para resolver las consultas se indicará una posición en el ráster o una ventana espacial y un rango temporal, los que representarán los limites de la sección a consultar. Por ejemplo, si se tiene una serie de tiempo de rásters de la temperatura de un lugar, se podrá consultar la temperatura de un determinado lugar en un instante de tiempo, o una sección del

lugar en un intervalo de tiempo.

1.2 Hipótesis

La hipótesis a validar con este trabajo de tesis es: *"si creamos una estructura de datos que aproveche no solo la localidad espacial, sino también la localidad temporal, entonces vamos a conseguir que nuestra estructura de datos ocupe menos espacio que almacenar cada ráster de manera compacta por separado."*

Para comprobar esta hipótesis nos mediremos en espacio de almacenamiento y tiempo de consultas, apuntando a tener mejores resultados en espacio de almacenamiento y tener resultados comparables en tiempo de consultas.

Nuestra comparación se realizará con una estructura que almacena cada ráster por separado y que sólo explota la localidad espacial de los rásters; apuntando a que nuestra estructura va a requerir utilizar menos espacio de almacenamiento.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar una versión eficiente en espacio de almacenamiento y tiempo de consulta de series de tiempo de rásters que tengan localidad temporal, adaptando estructuras existentes para luego compararla experimentalmente con otras estructuras empleadas para este dominio.

1.3.2 Objetivos Específicos

- Estudiar estructuras existentes para almacenamiento de rásters.
- Recolectar y adaptar datos de prueba que cumplan con el dominio de nuestro trabajo.
- Implementar y/o recopilar los códigos fuentes disponibles de los trabajos de investigación estudiados, correspondientes a estructuras que almacenan cada ráster por separado.

- Comparar los resultados obtenidos en cuanto a espacio y tiempo de consulta de nuestra propuesta con una estructura que almacene cada ráster de forma compacta por separado.

Capítulo 2

Discusión Bibliográfica

En esta sección se definirán conceptos que se utilizarán en este trabajo. Además se revisarán trabajos existentes que se enfocan en solucionar problemas similares al abordado.

2.1 Conceptos Previos

2.1.1 Sistemas de Información Geográfica

Un Sistema de Información Geográfica (SIG) [1] corresponde a una herramienta que realiza múltiples funciones, como almacenar, visualizar, consultar, analizar y modelar un conjunto de datos espaciales en formato digital, los que permiten representar características de una determinada superficie en un instante de tiempo, como las precipitaciones, la temperatura, altitud; o variables sociales, como el número de habitantes, densidad de población, entre otros. Hay dos maneras de almacenar los datos: modelo vectorial o modelo de ráster.

El modelo vectorial es representado mediante formas geométricas (puntos, líneas o polígonos) y el modelo de ráster divide el espacio en un conjunto regular de celdas, formando una matriz bidimensional con valores numéricos que representan alguna característica de la superficie en un determinado instante de tiempo, como por ejemplo, la temperatura de ese lugar en un determinado instante.

Ráster

Un ráster es una matriz que permite almacenar datos geográficamente referenciados. Como se puede observar en la figura 1.3 cada celda de la matriz bidimensional contiene valores que representan fenómenos del mundo real de algún plano: un mapa de temperatura, precipitaciones, densidad de población, etc. Los rásteres son fotografías

aéreas digitales, imágenes de satélite, imágenes digitales o incluso mapas escaneados.

Si bien la estructura de datos ráster es simple, es útil para una amplia variedad de aplicaciones. Las ventajas de almacenar los datos en forma de ráster son las siguientes:

- Estructura de datos simple: matriz de celdas con valores que representan una coordenada.
- Capacidad de representar superficies continuas y llevar a cabo análisis de superficie.
- Capacidad de almacenar puntos, líneas, polígonos y superficies de manera uniforme.

El tamaño de las celdas puede ser tan grande o pequeña como sea necesario para representar la superficie, por ejemplo, un kilómetro cuadrado, un pie cuadrado o incluso un centímetro cuadrado. El tamaño de celda determina el grosor o la fineza con la que aparecerán los patrones o las entidades en el ráster. Cuanto más pequeño sea el tamaño de celda, más suave o más detallado será el ráster. Sin embargo, cuanto mayor sea el número de celdas, más tiempo se tardará en procesar, aumentándose a su vez la demanda de espacio de almacenamiento. Si el tamaño de una celda es demasiado grande, se podría perder información o los patrones sutiles podrían oscurecerse. Por ejemplo, si el tamaño de celda es superior al ancho de una carretera, la carretera podría no existir en el dataset ráster.

Series de tiempo de rásters

Como se mencionó en la sección 2.1.1 un ráster representa información de un área determinada en un instante de tiempo. Estos datos pueden ir cambiando a través del tiempo, por lo que en muchos dominios se hace necesario tener la variación de estos datos en el tiempo. Para ello se utilizan varios ráster, donde cada uno representa la información en un determinado momento, teniendo así una serie de tiempo de rásters que representa una región en un intervalo de tiempo. Cabe mencionar que en cada

ráster existe localidad espacial, debido a que los datos cumplen con la primera ley de la geografía, es decir, los valores de celdas cercanas son similares entre sí. Además, un ráster que tiene información de un instante t_i, tendrá relación temporal con un ráster de un instante t_{i-1} y t_{i+1}, a esto le llamaremos localidad temporal. Todo lo anterior se traduce en que los datos de celdas cercanas en rásters próximos entre sí, son similares. Esta localidad espacio-temporal que cumplen las series de tiempo de rásters, es la propiedad que nuestra propuesta pretende explotar.

En la figura 2.1 se puede ver una serie de rásters que muestran los datos de una región en diferentes instantes de tiempo.

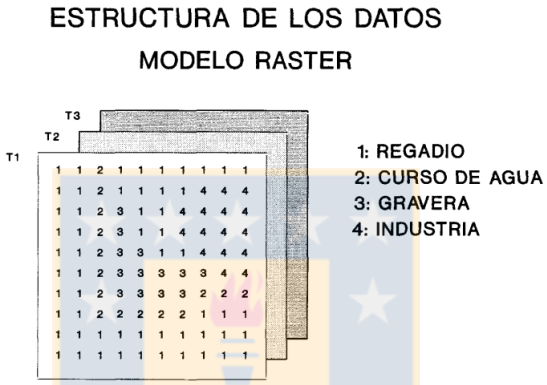

Figura 2.1: Varios ráster de un lugar en diferentes instantes de tiempo (Fuente [11]).

2.1.2 Curva de Orden Z

La curva de Orden Z [4] es una curva de llenado del espacio [3] que transforma un espacio de datos multidimensional en un espacio de una sola dimensión. Esto lo realiza manteniendo la localidad espacial existentes en los datos del espacio multi-dimensional. En la figura 2.2 se puede ver el comportamiento de esta curva, y como quedan los datos provenientes de un espacio bidimensional en un espacio de una sola dimensión.

Para encontrar un valor que estaba en el espacio bidimensional en el unidimensional, se utiliza el siguiente proceso.

Se toman los índices (x,y), en su representación binaria, y se intercalan los bits de una coordenada (x) con los bits de la otra (y). Primero el bit menos significativo

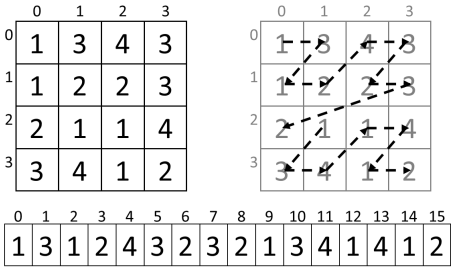

Figura 2.2: Comportamiento curva de Orden Z. (Fuente: Elaboración propia)

de la coordenada y, luego el primer bit menos significativo de la coordenada x pasará a ser el segundo bit menos significativo del nuevo valor (índice del arreglo), y así hasta considerar todos los bits. Como se muestra en la figura 2.3, para saber en qué posición del arreglo está el valor 4, el cual está ubicado en la posición (2,3) en el espacio bidimensional, se toman la representación binaria del 2 y 3, y se intercalan sus bits. Luego el valor formado corresponde al 13, por lo que en la posición 13 del arreglo se encuentra el valor 4.

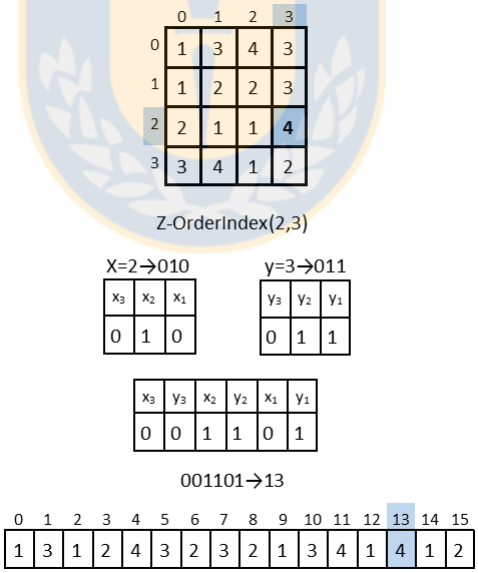

Figura 2.3: Cálculo de ZOrderIndex, para la coordenada (2,3). (Fuente: Elaboración propia)

2.1.3 k^2-*tree*

Un k^2-tree es una estructura de datos diseñada para la representación de matrices esparsas binarias. Fue originalmente diseñado como una representación comprimida de grafos Web, representando su matriz de adyacencia en un espacio reducido. Luego, ha sido aplicado para la compresión de otros dominios.

La raíz del k^2-tree corresponde a la matriz completa de tamaño $n \times n$. Luego la matriz es particionada en k^2 submatrices de tamaño $\frac{n}{k} \times \frac{n}{k}$. Las k^2 submatrices son tomadas de izquierda a derecha y de arriba abajo; un nodo hijo es agregado a la raíz por cada uno de ellas. Cada nodo del árbol es etiquetado con un bit: 1 si la submatriz contiene al menos un 1 o 0 en otro caso. Luego el proceso continúa recursivamente solo por las submatrices que contienen al menos un uno. La subdivisión recursiva de la matriz termina cuando la submatriz actual está completa de ceros o cuando las celdas de la matriz original son alcanzadas. La figura 2.4 muestra una matriz binaria con su representación en un k^2-tree, donde se puede notar que los nodos que no tienen hijos, representan a las submatrices que sólo contienen 0s.

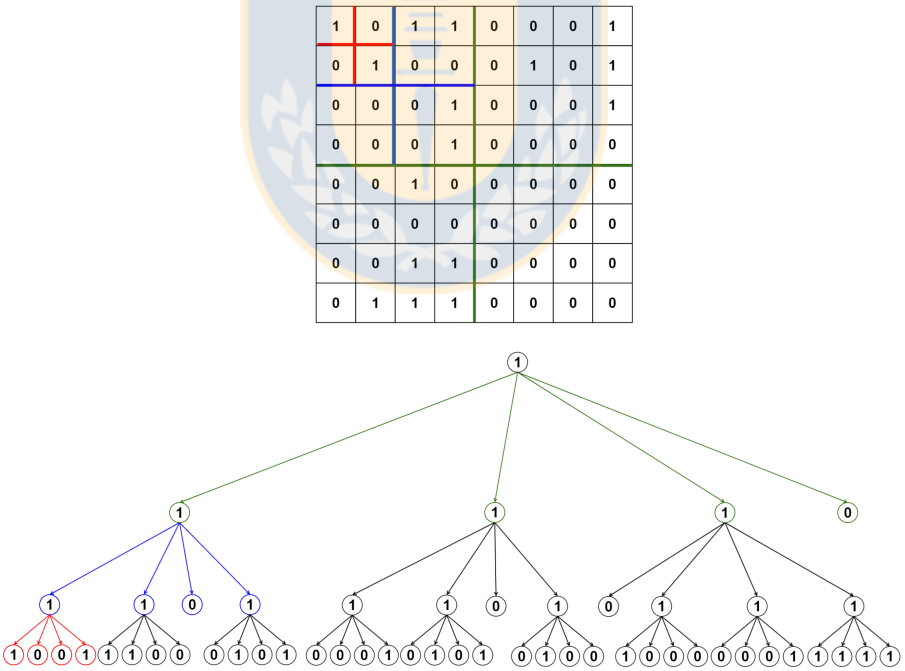

Figura 2.4: Matriz bidimensional y su k^2-tree (Fuente [5]).

El árbol formado es representado por niveles con 2 bitmaps, uno que representa

T= 1 1110 1101 1101 0111 1001 1100
L = 0101 0001 0101 0100 1000 0001 1111

Figura 2.5: Bitmaps del k^2-tree de la figura 2.4 (Fuente [5]).

todos los niveles excepto el último y el segundo representa el último nivel. En la figura 2.5 se ven los dos bitmaps que representan el k^2-tree de la figura 2.4

2.1.4 k^3-*tree*

El concepto del k^2-tree es facilmente generalizable a más dimensiones. En particular para 3 dimensiones, el k^3-tree es una estructura que dado un valor k y un cubo de datos de lado potencia de k, divide el cubo en k^3 subcubos, los cuales son representados con un 0s o un 1s dependiendo de si en el subcubo hay 1s o no. A los sub-cubos que fueron representados con un 1, se les aplica el procedimiento recursivamente, generando así un árbol.

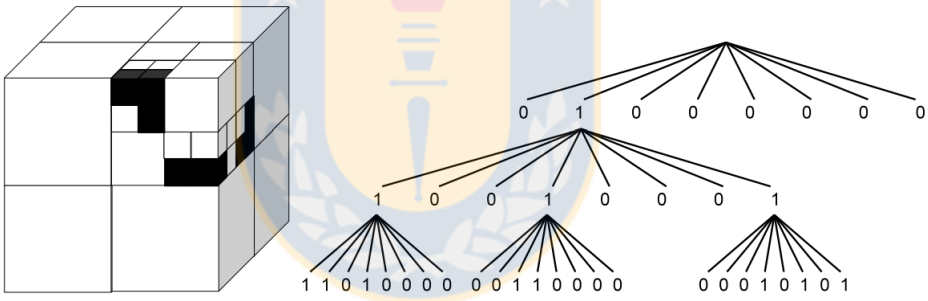

Figura 2.6: Matriz binaria tridimensional y su k^3-tree (Fuente [7]).

La figura 2.6 muestra un ejemplo de un k^3-tree basado en una matriz tridimensional, que es representada como un cubo vacío, en el cual las celdas que contienen el valor 1 son pintadas en negro. El nodo raíz del árbol conceptual tiene 8 hijos, correspondientes a las 8 submatrices de la partición inicial. Solo la segunda submatriz tiene al menos un 1 (región en negro), por lo que es subdividida. El orden de la representación de las submatrices en los nodos está representada en la figura 2.7.

El k^3-tree es almacenado usando los mismos bitmaps que en el k^2-tree; un bitmap almacena los bits de todos los niveles excepto el último y otro bitmap almacena el

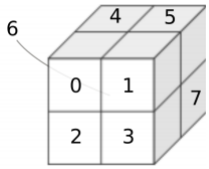

Figura 2.7: Orden de submatrices para los nodos del k^3-tree (Fuente [7]).

último nivel. En la figura 2.8 se tiene los dos bitmaps que representan al k^3-tree mostrado en la figura 2.6.

```
T: 01000000 10010001
L: 11010000 00110000 00010101
```

Figura 2.8: Bitpams del k^3-tree (Fuente [7]).

2.2 Trabajo Relacionado

Un trabajo previo al nuestro, en el cual se proponen estructuras de datos para representar rásters aprovechando su localidad espacial, es [5], la que describe varias alternativas de solución. En particular, esta tesis es la generalización de una de las propuestas de ese trabajo. A continuación se describen estas alternativas presentadas en dicho trabajo.

2.2.1 Primera alternativa

Esta alternativa contiene varios pasos:

- **Mapeo de 2 dimensiones a una dimensión:** Para realizar el mapeo 2D1D toma un ráster (bidimensional) y mediante una curva de llenado del espacio, en particular la curva de orden Z, lo convierte en una secuencia de valores (unidimensional).
- **Crear árbol binario:** La secuencia obtenida en el paso anterior es convertida en un árbol binario. Este es creado tomando el centro de la secuencia como raíz,

la subregión izquierda como subárbol izquierdo y la subregión derecha como subárbol derecho.

- **Reducción de magnitud:** Para reducir la magnitud de los valores de los nodos del árbol, se calcula la diferencia entre el valor almacenado en un nodo y su nodo padre. Debido a que la representación de números negativos es bastante costosa (la secuencia de bits a la izquierda es de 1s) se codifican los valores de los nodos usando *ZigZag Encoding*, técnica que convierte valores enteros a valores enteros no negativos.

- **Podar los subárboles:** Lo siguiente es podar los subárboles con el mismo valor, convirtiendo el árbol binario completo en un árbol sucinto donde cada subárbol tiene 0 o 2 hijos. Este nuevo árbol podado se representa con 2 secuencias; una con los valores de los nodos almacenados por niveles y otra con un 0 o 1 que indica si un nodo tiene 0 o 2 hijos.

- **Codificación de valores:** La secuencia de valores es codificada usando códigos de acceso directo (*Direct Access Codes*, DACs) ya que son valores enteros pequeños no negativos. La secuencia de 0s y 1s es codificada con alguna técnica que soporte operaciones de *Rank* y *Select* sobre bitmaps.

Esta estructura soporta consultas de acceso y consultas por subregión.

2.2.2 Segunda alternativa

Esta alternativa consiste en un mapeo de 3 dimensiones a 2 dimensiones (3D2D). Esta estructura toma como coordenadas (x,y,z), donde x e y son las filas y columnas respectivamente, y z el valor almacenado en dichas coordenadas. Las primeras 2 coordenadas son mapeadas con la curva de Orden Z para formar un nuevo eje de coordenadas, donde P_{xy} es el valor generado por la curva de Orden Z, con x e y la fila y columna respectivamente. Por otro lado, el rango de posibles valores se toma como eje de coordenadas, de valores v. En esta representación, en una coordenada (P_{xy}, v) habrá un 1, si en en el ráster en la fila x y columna y esta almacenado el valor v.

Al aplicar este mapeo, se obtiene una grilla binaria, la cual se comprime con un k^2-tree y todas las consultas son mapeadas a consultas sobre el mismo, incluyendo la consulta por subregión con rango de valores restringido, que no soporta la primera alternativa.

Ambas alternativas presentadas, permiten representar solo un ráster y, por tanto, no aprovecha la localidad temporal de una serie de tiempo de rásters. Una alternativa para representar series de tiempo de rasters, es aplicar una de estas estructuras recién descritas a cada raster de la serie; sin embargo, en este trabajo mostramos que es posible reducir todavía más el espacio aprovechando la localidad temporal existente en la serie. Nuestro trabajo se basa en la segunda alternativa de las antes descritas ya que, si bien sus resultados suelen ser ligeramente peores, su generalización a un espacio de 3 dimensiones resulta natural.

2.2.3 Otras estructuras de datos compactas para rásters

En [12] también se propusieron varias variantes del $k^2 - tree$ y se demostró su utilidad para superar los enfoques clásicos como GeoTIFF. Un trabajo más reciente [13] propuso el k^2-raster. La primera parte de esta estructura es una variante del $k^2 - tree$ que realiza el mismo tipo de particionamiento pero que se detiene al llegar a submatrices del ráster en las cuales todas las celdas contienen el mismo valor. Es decir, los 1s en esta representación significan que la submatriz correspondiente contiene 2 o más valores diferentes, mientras que los 0s significan que toda la submatriz contiene el mismo valor. En paralelo a esta estructura se guarda un árbol de valores mín/máx que almacena, por cada submatriz el valor mínimo y máximo de dicha submatriz en el ráster. Dichos valores no se almacenan en plano sino que se realiza una codificación diferencial. Esta representación soporta los tres tipos de consulta (access, windowQuery y rangeQuery). Los algoritmos que soportan dichas consultas se basan en recorridos de la raíz a las hojas tanto en k^2-treee como en el árbol de diferencias min/max. La evaluación experimental realizada por los autores muestra un comportamiento similar al estado del arte cuando el raster tiene pocos valores, pero una mejora drástica cuando el número de valores aumenta. Las mejoras se producen tanto en espacio de almacenamiento como en tiempo de consulta.

A comienzos de este año se presentó [14], la cual es una generalización a dicha estructura para series de tiempo de ráster. Esta estructura se basa en una técnica conocida en otros dominios, como objetos en movimiento, que almacena snapshots (utilizando un k^2-ráster) y vitácoras de diferencias. A diferencia de nuestro trabajo, dicha estructura no soporta consultas de rango espacial.

Capítulo 3

Solución Desarrollada

3.1 Nuestra Solución

Debido al gran volumen de información que generan los rásters en un SIG, más aún cuando se tiene la variación de estos datos en el tiempo, nuestra propuesta de solución fue crear una representación para series de tiempo de ráster con localidad temporal. Para ello, extendimos la propuesta mencionada en la sección 2.2 la cual está originalmente pensada para solo un ráster y no series de estos con localidad temporal. De manera esquemática, la idea consiste en que, dada una serie de tiempo de rásters, aplicar la curva de orden Z a cada ráster de la serie, creando una secuencia de valores para cada ráster del conjunto. Luego, crear una serie de matrices binarias en base a cada una estas secuencias. Cada matriz se forma considerando en un eje la curva de Orden Z y en el otro eje los valores posibles. Finalmente, se compacta el conjunto de matrices binarias utilizando un k^3-tree.

Aparte de estar espacialmente juntos los 1s por la localidad espacial que tiene cada ráster, la localidad temporal hace que estos estén cerca también en la tercera dimensión, lo que facilita su compactación.

En el ejemplo de la figura 3.1, tomando el ráster de $T1$, se recorre los valores en *Z-Order* pasando a una dimensión reflejada en $T1$ de la figura 3.2. A partir de cada una de estas secuencias de valores es formada una matriz binaria para cada una de ellas. Considerando el ejemplo, los primeros cuatro valores del ráster $T1$ en *Z-Order* son 1, 0, 3 y 1, para los cuales en la matriz binaria de la figura 3.3 se completa con un 1s en la fila 1 de la primera columna, en la fila 0 de la segunda columna, en la fila 3 de la tercera columna y en la fila 1 de la cuarta columna. Las demás celdas de estas cuatro columnas son completadas con 0s. Similarmente, el primer valor del ráster $T2$ es 1, por lo que en la matriz binaria de la figura induce a tener un 1 en la columna 1, fila 1.

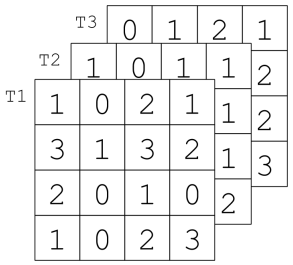

Figura 3.1: Serie de tiempo de rásters. (Fuente: Elaboración propia)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1	1	0	3	1	2	1	3	2	2	0	1	0	1	0	2	3
T2	1	0	2	1	1	1	2	1	3	1	2	1	2	1	3	2
T3	0	1	3	2	2	1	1	2	3	0	1	0	3	2	2	3

Figura 3.2: Resultado de la curva de Orden Z para cada ráster de la figura 3.1. (Fuente: Elaboración propia)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0
2	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0
3	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1

Figura 3.3: Matriz binaria para la secuencia T1. (Fuente: Elaboración propia)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0
2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0

Figura 3.4: Matriz binaria para la secuencia T2. (Fuente: Elaboración propia)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0
2	0	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0
3	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1

Figura 3.5: Matriz binaria para la secuencia T3. (Fuente: Elaboración propia)

3.2 Consultas

Teniendo la representación para el conjunto rásters, definimos 4 tipos de consultas que nos permitirán acceder a los datos del dominio abordado.

3.2.1 Acceso

Definimos una consulta de acceso como *acceso(x,y,t)*, la cual nos permite acceder al valor registrado en una columna *x*, una fila *y* en el ráster *t*, en términos generales, corresponde al valor registrado en coordenada de posición *x,y*, en un instante de tiempo *t*. Ejemplo, la temperatura en Concepción el 25 de diciembre de 2017 a las 18:00.

Algoritmo 1 acceso(x, y, z), consulta de acceso en el k^3*-tree*

Entrada: x, y, las coordenadas de la celda en la matriz, z ráster.
Salida: v el valor en la coordenada solicitada.

```
1: p_{x,y} ← z_order_trans(x, y)
2: list ← get_inverse_list(k3tree, p_{x,y}, z)
3: v ← list.element()
4: return v
```

En el Algoritmo 1 muestra la implementación de la consulta de acceso, donde la función $get\_inverse\_list$ del k^3-tree recupera los 1s encontrados en columna de la matriz binaria donde está el valor buscado, que en nuestro caso, es un único 1 por columna que se almacena en la matriz binaria.

3.2.2 Ventana

Una consulta de ventana es definida como *Window(x1,y1,x2,y2,t1,t2)*, la que nos permite obtener todos los valores registrados en una zona restringida (*x1,y1,x2,y2*) durante un intervalo de tiempo (*t1,t2*). Por ejemplo, la temperatura en la Octava región entre septiembre y noviembre de 2018.

La estrategia de consulta consiste en realizar primero una descomposición en quadboxes [15] [16] y luego, por cada quadbox, realizar una consulta de rango en el k^3-tree. La descomposición en quadboxes es necesaria debido a que la transformación mediante *Z-Order* no produce necesariamente un único rango contiguo en una de las dimensiones de la matriz.

3.2.3 Rango

La definimos como *rango(r1,r2)*. Esta consulta nos permite obtener los lugares e instantes de tiempo en donde los valores están en un intervalo definido entre *r1,r2*.

Algoritmo 2 Ventana($x1, y1, x2, y2, z1, z2$), con estrategia de descomposición en $quadboxes$.

Entrada: $x1, y1, x2, y2$ las coordenadas de la subregión de consulta, $z1, z2$ intervalo de rásters.

Salida: M matriz de valores que conforma la subregión consultada.

```
1:  M ← null
2:  for k ← z1 to z2 do
3:     Q ← quadbox_desc(x1, y1, x2, y2)
4:     while not Q.isEmpty() do
5:        quad ← Q.element()
6:        node ← acceso(quad.x1, quad.y1)
7:        p_end ← z_order_trans(quad.x2, quad.y2)
8:        while node.position ≠ p_end do
9:           M.insert(node)
10:          node ← node.next()
11:       end while
12:       quad ← Q.next()
13:    end while
14: end for
15: return M
```

Ejemplo, lugares con sus respectivas fechas donde la temperatura estuvo entre 25 y 30 grados.

3.2.4 Ventana con Rango Restringido

Es definida como *WindowRange(x1,y1,x2,y2,t1,t2,r1,r2)*, la cual representa una consulta de ventana con parámetros *x1,y1,x2,y2,t1,t2* con el rango restringido de valores posibles a *(r1, r2)*. Notar que si *r1* es el valor mínimo y *r2* es el valor máximo, esta consulta es idéntica a una consulta de Ventana. Por ejemplo, días calurosos en la Octava Región, entre el 29 de febrero y el 31 de marzo, donde la temperatura estuvo entre 30 y 35 grados.

Esta consulta, se implementa directamente como una consulta de rango en el k^3-tree ya que, a diferencia de la anterior, la dimensión espacial es consultada en toda su extensión, por lo que no se requiere descomposición en quadboxes.

Capítulo 4

Evaluación Experimental

4.1 Implementación

El marco experimental esta dado por una máquina de las siguientes características:

- Procesador Intel Core i7-3820@3.60GHz.
- Memoria RAM de 32GB.
- Sistema operativo Ubuntu server (kernel 3.13.0-35).
- Compilador gnu/g++ versión 4.6.3.

4.2 Dataset

Para nuestros experimentos, usamos conjuntos de datasets de temperaturas reales entre Septiembre y Octubre de 2017. Este dominio fue elegido debido a que cumple con la localidad espacial y temporal de los datos, ya que las temperaturas registradas en zonas cercanas deberían ser similares entre sí, de igual manera para la localidad temporal, ya que en una misma zona, la temperatura va cambiando en el tiempo de manera continua.
Los datasets fueron descargados [1] diariamente, durante dos meses. Las localidades estudiadas son Puerto Rico, Guam y Hawaii, para cada una de estas recortamos rásters de 64x64 y 128x128, para poder tener datasets de diferentes tamaños. Además de estos, utilizamos de tamaño 193x193 para el caso de Guam, el cual además generamos una versión promediada por día, esto fue realizado para medir el comportamiento de la estructura al disminuir la localidad temporal de los datos, ya que al promediar la temperatura diaria, que además tiene un tiempo de muestreo de 3 horas,

[1] http://www.ftp.ncep.noaa.gov/data/nccf/com/rtma/prod/

se espera a que los valores no sean tan similares como cuando el tiempo de muestreo es más frecuente. La tabla 4.1 muestra el detalle del número de columnas, filas y de rásters para cada datasets. Los datasets de Puerto Rico y Hawaii son muestras de tiempo tomadas cada 1 hora, a diferencia de los datos de Guam que fueron capturados cada 3 horas. Debido a esto, el número de rásters para Guam es menor que los otros datasets.

Dataset	Columnas	Filas	Rásters
Guam-64	64	64	456
Hawaii-64	64	64	1368
Puerto-Rico-64	64	64	1368
Guam-128	128	128	456
Hawaii-128	128	128	1368
Puerto-Rico-128	128	128	1368
Guam-193	193	193	456
Guam-diario-193	193	193	62

Tabla 4.1: Datasets

Cabe destacar, que la frecuencia con la que se toman los datos, afecta directamente la localidad temporal de los datasets. Por lo que debido a que en Hawaii y Puerto Rico la frecuencia de muestreo es cada una hora, tiene una mejor localidad temporal que los dataset de Guam.

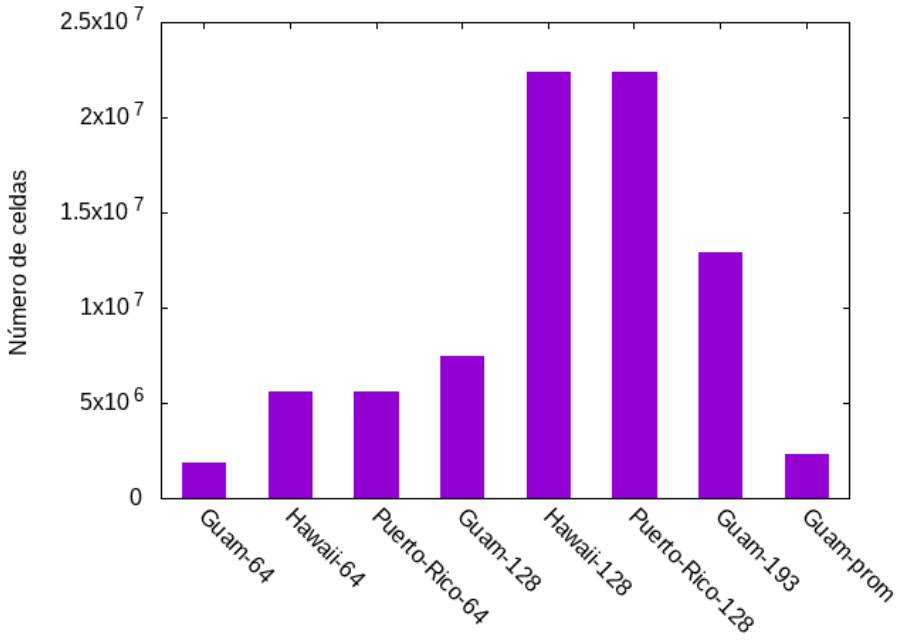

Figura 4.1: Número de celdas para cada dataset

4.3 Experimentos

Los experimentos fueron realizados para medir el uso de memoria de las estructuras y de los tiempos de consultas definidas en las secciones anteriores para cada estructura.

4.3.1 Uso de Memoria

En la figura 4.2 se muestra el tamaño en bytes del dataset original, representado con k^2-tree y con k^3-tree, donde k^2-tree es aplicar a cada ráster de la serie la propuesta descrita en [5], mientras que k^3-tree es nuestra propuesta de considerar la serie de tiempo de ráster completa y no por separado.

Además de medir el espacio utilizado por cada estructura, consideramos el número de bits que se utiliza por celda del datasets. Estos resultados se pueden ver en la tabla 4.2 donde se calculó el número total de bits dividido por el total de celdas de cada datasets. El total de celdas corresponde a $columnas \times filas \times rasters$.

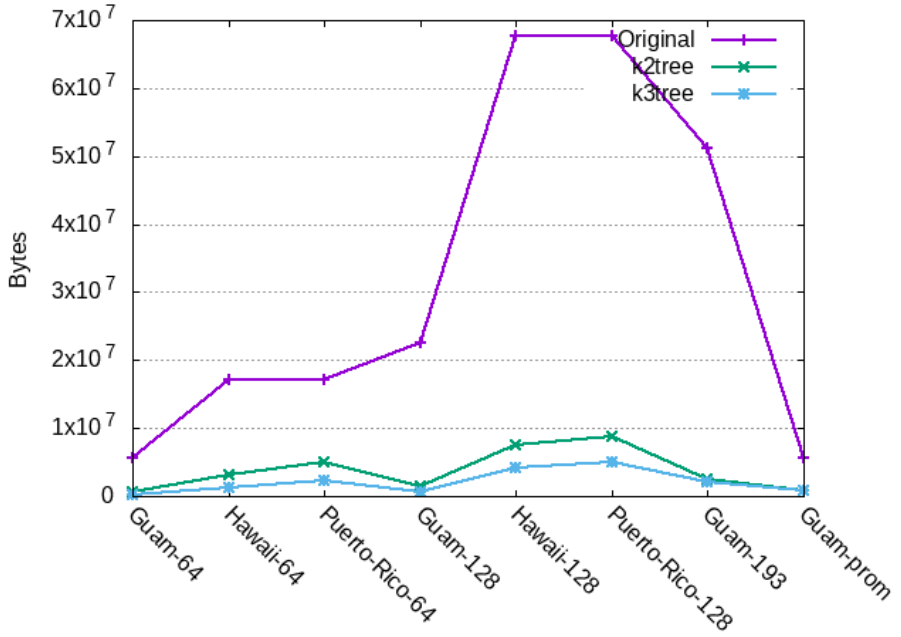

Figura 4.2: Espacio utilizado en bytes

Dataset	Espacio Total (MB)			Bits por celda	
	Original	k^2-tree	k^3-tree	k^2-tree	k^3-tree
Guam-64	5.44	0.65	0.17	2.9	0.8
Hawaii-64	16.33	2.90	1.12	4.3	1.7
Puerto-Rico-64	16.34	4.70	2.20	7.0	3.3
Guam-128	21.50	1.33	0.65	1.5	0.7
Hawaii-128	64.50	7.24	4.01	2.7	1.5
Puerto-Rico-128	64.52	8.29	4.82	3.1	1.8
Guam-193	48.77	2.46	2.09	1.6	1.4
Guam-diario-193	5.37	0.75	0.81	2.7	2.9

Tabla 4.2: Espacio utilizado

Localidad temporal

Para medir la compresión que se alcanza aprovechando la localidad temporal de los ráster, se comprimió el dataset de Hawaii pero tomando espacios de tiempos distintos, esto es, cada 1, 3, 6, 12 y 24 horas. En el gráfico 4.3 se puede ver la comparativa entre ambas estructuras, aparte de presentar mejores resultados que el $k^2 - tree$ muestra una tendencia que mientras más cercanos temporalmente sean los rásters hay una

mejor tasa de compresión, esto debido a la localidad temporal de los rásters cercanos.

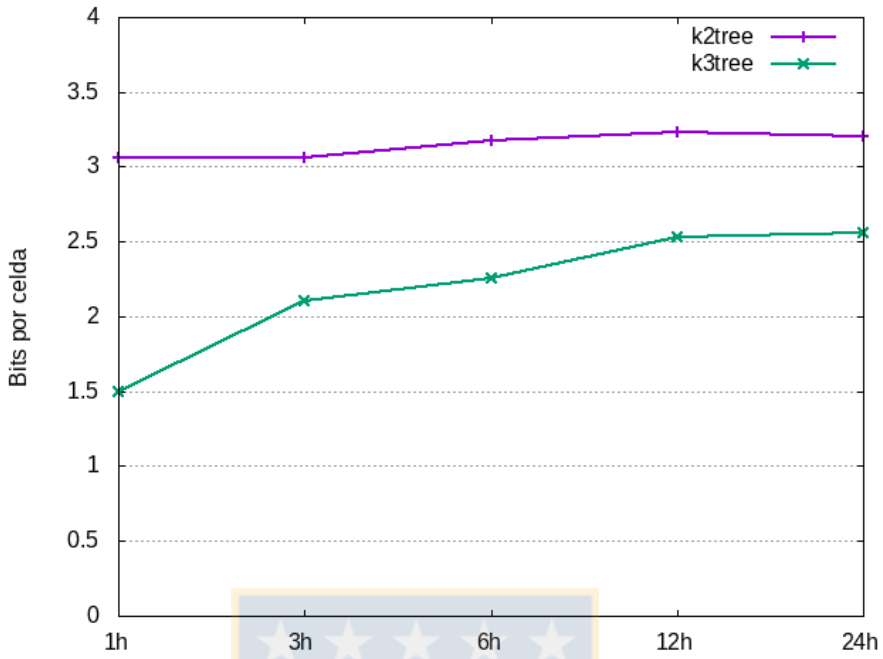

Figura 4.3: Espacio utilizado en bytes

De lo anterior, nuestra hipótesis queda validada, pues nuestros experimentos muestran que se puede ahorrar hasta un 74% de espacio usando una estructura de datos compacta que explota tanto la localidad espacial como la localidad temporal de la serie de tiempo de rásters frente a una estructura de datos compacta que explota solo la localidad temporal. Cabe mencionar que dado que el dataset Guam Diario tiene un espacio muestral de 3 horas que además fue promediado por día, perjudica su localidad temporal, de modo que nuestra estructura no puede explotar dicha propiedad lo que se ve reflejado en la tabla 4.2.

4.3.2 Tiempos de Consultas

Acceso

Las consultas de acceso para nuestro dominio se podría interpretar como ¿Cuál es la temperatura que hay en un determinado lugar en un instante de tiempo?. Para nuestros experimentos se ejecutaron 10.000 consultas 50.000 veces, esta repetición

se realizó para que los resultados no estén afectados por el “ruido” de la máquina. Cabe destacar que una consulta de acceso para el k^3-tree sería *acceso(x,y,t)* la cual es mapeada para el k^2-tree de la forma *acceso(x,y)* en el ráster *t*. La tabla 4.3 muestra el tiempo de consulta de acceso para cada dataset.

Dataset	k^2-tree	k^3-tree
Guam-64	0,62	2,20
Hawaii-64	0,58	2,37
Puerto-Rico-64	0,60	2,71
Guam-128	0,62	2,46
Hawaii-128	0,62	2,42
Puerto-Rico-128	0,62	2,54
Guam-193	0,62	2,26

Tabla 4.3: Tiempos por consulta de Acceso para el k^2-tree y k^3-tree en (us)

La figura 4.4 muestra un gráfico con los tiempos totales de las consultas de acceso, donde el item “Total k2” es el tiempo que tarda en cargar los ráster y consultar, mientras que “total-query-k2” es solo el tiempo que tarda en realizar las consultas (lo mismo para k3). En el gráfico se puede ver que ambos tiempos del k^3-tree son muy similares entre sí a diferencia del k^2-tree, lo que nos muestra que hay una diferencia entre cargar cada ráster por separado y realizar consultas ($k^2 - tree$) con respecto a cargar la estructura completa sólo una vez y ejecutar las consultas ($k^3 - tree$).

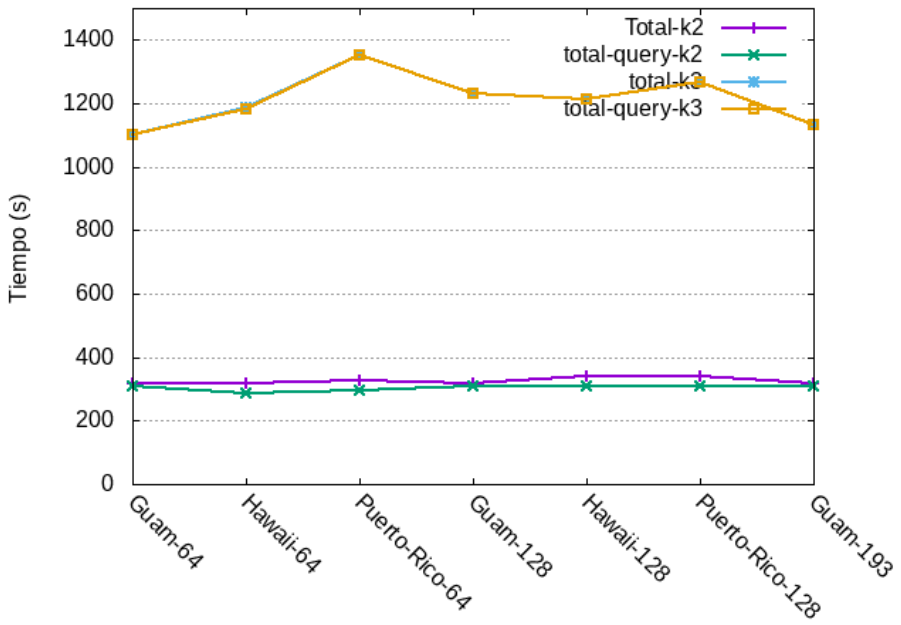

Figura 4.4: Tiempo total de consulta de acceso

Ventana

La consulta de ventana, en nuestro dominio tendría como finalidad obtener todas las temperaturas registradas en una zona limitada. En este caso se ejecutaron 5.000 consultas 3 veces, con ventanas de tamaño 10×10×10, 20×20×20, 50×50×50 y 100×100. Es importante mencionar, que las 5.000 consultas fueron mapeadas para poder consultarlas a la estructura del k^2-tree, lo que implica que a éste se le consultaron a lo menos 5.000 consultas. Esto se debe a que, una consulta de ventana del estilo *ventana(0,0,5,5,1,3)* se transforma en 3 consultas para el k^2-tree, del tipo *ventana(0,0,5,5)* la cual debe ser consultada al ráster 1, ráster 2 y ráster 3.

Tomando en consideración el dataset más grande en cantidad de celdas, Hawaii-128, en la figura 4.5 nos muestra el tiempo en microsegundos que tarda cada estructura en cada consulta con diferentes tamaños de ventanas. Se puede observar que nuestra solución presenta mejores resultados con ventanas de tamaño pequeño, ya que cuando la ventana es de 100×100×100 la estructura del $k^2 - tree$ presenta mejor tiempo de consulta.

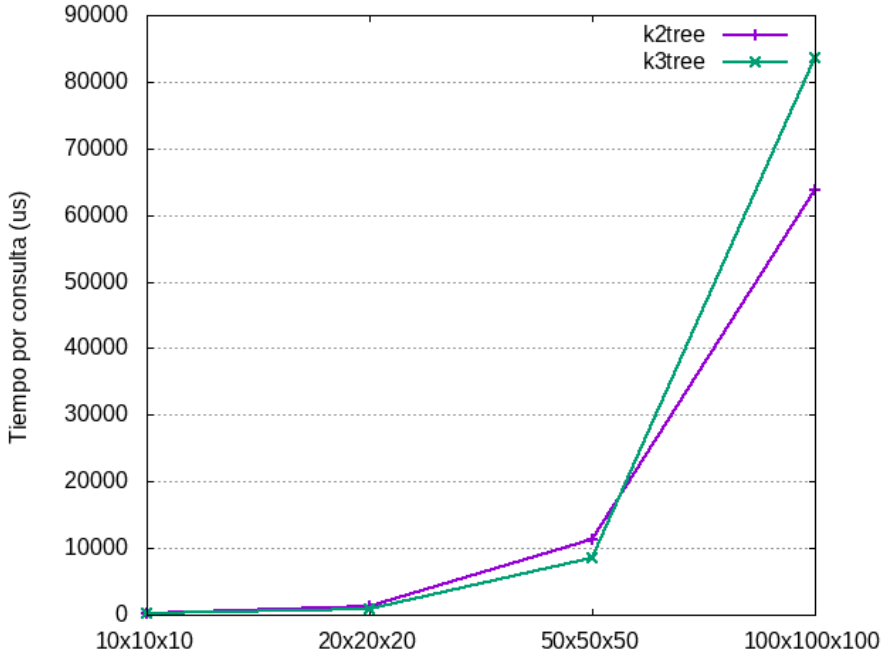

Figura 4.5: Tiempo por consulta de Ventana para Hawaii-128

Ventana con Rango Restringido

Una interpretación de esta consulta para nuestro dominio de estudio sería obtener todas las temperaturas que están dentro de una zona restringida y que estén dentro de un rango de consulta. Para ese experimento se probó 5.000 consultas 3 veces, para ventanas de tamaño $10\times10\times10$, $20\times20\times20$, $50\times50\times50$ y 100×100, con un rango de tamaño 10, 20, 50. Considerando el dataset de Hawaii-128, la figura nos muestra el tiempo por consulta con un rango fijo de 20 para diferentes tamaños de ventanas. 4.6

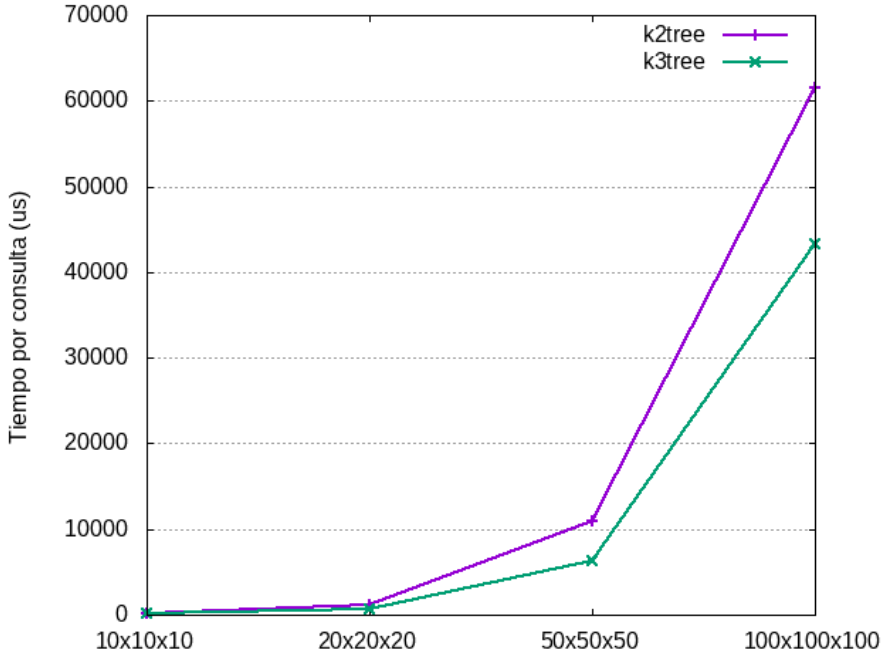

Figura 4.6: Tiempo por consulta con rango 20

4.4 Discusión de los Resultados

4.4.1 Espacio

Los resultados presentados del espacio utilizado por cada estructura en la figura 4.2 nos muestra que ambas estructuras tienen un buen comportamiento con respecto al espacio utilizado por el datasets original. Sin embargo, este es el resultado principal de este trabajo de tesis ya que valida la hipótesis planteada al mostrar que la estructura formada por el k^3-tree utiliza menos espacio que la estructura del k^2-tree, demostrando que si utilizamos una estructura que aproveche no solo la localidad espacial, sino también la localidad temporal, vamos a conseguir que ocupe menos espacio que almacenar cada ráster de manera compacta por separado, lo que se ve reflejado en el experimento realizado con diferentes tiempos de muestreo, ya que tiene una mejor tasa de compresión cuando la frecuencia es cada 1 hora en comparación a cuando es cada 24 horas.

4.4.2 Consultas

En cuanto a los resultados obtenidos en las consultas, se puede observar que, hay una diferencia significativa para el k^2-tree al momento de consultar varios ráster, ya que a diferencia del k^3-tree, éste carga una sola estructura y una sola vez. En cambio el k^2-tree debe estar cargando constantemente cada ráster que desea consultar. Esto se ve reflejado en la figura 4.4 donde ambas curvas del k^3-tree son relativamente iguales, a diferencia de las curvas del k^2-tree donde se presenta una diferencia entre el tiempo que considera solo lo que tardan las consultas con respecto al tiempo que tarda completamente el proceso de consultar (cargar estructura y realizar consultas).

Capítulo 5

Conclusiones y Trabajo Futuro

La presente tesis propone una nueva estructura de datos que permite aprovechar la localidad espacial y temporal de una serie de tiempo de rásters. A partir de un conjunto de experimentos realizados utilizando un dataset real de rásters de temperaturas, los cuales, cumplen con la propiedad de tener localidad espacial y temporal, obtuvimos resultados que muestran que nuestra estructura es capaz de comprimir utilizando un mínimo de 0.7 bits por celda versus a los 1.5 obtenidos por la estructura propuesta en el trabajo anterior frente al mismo dataset la cual solo aprovecha la localidad espacial de los datos. Este patrón se da en la mayoría de los casos de los experimentos realizados en diferentes escalas. Si bien dentro de nuestros experimentos existe un caso de fallo, es decir, nuestra estructura tiene un resultado ligeramente peor (2.9 bits por celda) frente a la estructura del trabajo anterior (2.7 bits por celda), esto se puede explicar debido a que ese conjunto de ráster resulta ser un promedio diario de temperaturas, lo que afecta directamente la localidad temporal de los datos. En cuanto a tiempos de consultas, nuestra estructura es competitiva frente a los resultados de la estructura del trabajo previo, sin considerar el hecho de que esta última en los experimentos no considera el tiempo de buscar el ráster consultado. En definitiva nuestro trabajo, presenta una alternativa viable para este tipo de compresiones proveyendo un conjunto de consultas relevantes para el dominio donde ésta sea útil. Como trabajo futuro queda por probar como es que nuestra estructura se comporta frente a otros trabajos que aprovechan la localidad espacial y temporal de los datos.

Bibliografía

[1] Worboys, Michael, and Matt Duckham. (2004) GIS: a computing perspective.

[2] Tobler, W. R (1970). A computer model simulation of urban growth in the Detroit region. Economic Geography. p.236.

[3] Sagan, Hans (1994), Space-Filling Curves

[4] Morton, G. M. (1966), A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing

[5] P. Alejandro Pinto. Representaciones compactas de matrices con localidad espacial. Tesis de Magíster, Universidad de Concepción.

[6] Rajeev Raman, Venkatesh Raman, and S. Srinivasa Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 233-242. SIAM Press, 2002.

[7] Guillermo de Bernardo Roca. New data structures and algorithms for the efficient management of large spatial datasets. Tesis Doctoral, Universidad de Coruña.

[8] url:http://eddytorresleon.blogspot.cl/2012/10/
Accedido por última vez 22 de Junio 2017.

[9] url:https://books.google.cl/books?id=oFRBCgAAQBAJ
Accedido por última vez 22 de Junio 2017.

[10] url:http://www.cobolca.com/2010/01/
Accedido por última vez 22 de Junio 2017.

[11] url: http://revistas.ucm.es/index.php/AGUC/article/viewFile/AGUC9595120011A/31635
Accedido por última vez 22 de Junio 2017.

[12] de Bernardo, Guillermo and Álvarez-García, S. and Brisaboa, N. R. and Navarro, G. and Pedreira, O. Compact Querieable Representations of Raster Data.

[13] Ladra, S. and Paramá, J. R. and Silva Coira, F.Compact and queryable representation of raster datasets

[14] Ana Cerdeira-Pena and Guillermo de Bernardo and Antonio Fariña and José R. Paramá and Fernando Silva-Coira- Towards a Compact Representation of Temporal Rasters.

[15] Proietti, G. (1999). An optimal algorithm for decomposing a window into maximal quadtree blocks. Acta Informatica, 36(4), 257-266.

[16] Tsai, Y. H., Chung, K. L., & Chen, W. Y. (2004). A strip-splitting-based optimal algorithm for decomposing a query window into maximal quadtree blocks. IEEE Transactions on Knowledge and Data Engineering, 16(4), 519-523.