USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

FACULTADE DE MATEMÁTICAS

Trabajo Fin de Master

Clústering Espacial Orientado a la Astronomía

Jorge Gómez Crespo

2012-2013

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

MASTER EN MATEMÁTICAS

Trabajo Fin de Master

Clústering Espacial Orientado a la Astronomía

Jorge Gómez Crespo

2012-2013

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Conocimiento: Astronomía y Astrofísica
Título: Clústering Espacial Orientado a la Astronomía
Director/a: José Ángel Docobo Durantez
Breve descrición del contenido
Los mecanismos de clústering han propiciado en los últimos tiempos importantes logros en múltiples ramas del saber. Habiendo detectado cierto vacío en su aplicación a la rama de la Astronomía, hemos buscado introducir los mecanismos del clústering de localización, llamado también espacial, dentro de esta especialidad científica con el objetivo de aplicarlo a grandes bases de datos en cuanto sea posible.
Recomendaciones
Otras observaciones

Índice general

Resumen

La minería de datos surge con el fin de estudiar y conocer mejor la incesante cantidad de datos obtenidos en multiples campos del saber, tanto de Humanidades como científicos, permitiéndonos explotarlos, y haciendo posible sorprendentes avances. En este ámbito, ha sido nuestra intención desarrollar un nuevo algoritmo de clústering, basado en el etiquetado dado por la curva de Hilbert y orientado a servir como una poderosa herramienta aplicada a las actuales necesidades, en este caso, astronómicas.

La presente contribución está estructurada siguiendo las características propias de un trabajo de fin de master en Matemáticas, es por ello, que comenzamos introduciendo los conceptos básicos utilizados en minería de datos y, en particular, los referentes a las herramientas de clústering. Exponemos también las principales curvas que llenan el espacio, su construcción y aplicaciones. Seguidamente, pasamos a describir el algoritmo y aportamos su pseudocódigo y una aplicación a un caso ficticio.

Finalmente, describimos como se llevaría a cabo el proceso de ZOOM, que se realizará en la parte de evaluación y refinamiento de los resultados. Así mismo, comentamos las venideras aplicaciones que queremos realizar en el campo de la astronomía y en especial aquellos que deseamos comenzar, e incluso, completar el próximo año académico. En la sección de Conclusiones, explicamos la importancia de este trabajo.

Por último, en el Apéndice, hacemos una recopilación de información de los catálogos OARMAC, WDS, HIPPARCOS y TYCHO, con los que ya estamos familiarizados y a los que deseamos aplicarles el algoritmo propuesto.

Palabras clave: Astronomía; Clústering; Curva de Hilbert que llena el Espacio (HSFC).

Abstract

Data Mining is centered on the study and the understanding of the increasing amount of data obtained in multiple knowledge fields, both in Humanities and Science, allowing us exploit the full potential of them and to accomplish surprising advances. With these terms of reference, our intention was to develop a new clustering algorithm based on the Hilbert Curve Index and applied as a powerful tool relative to the current requirements, in this case, astronomical ones.

This contribution is structured from a Master's final project point of view. Thus we began explaining the principal topics of the Data Mining field and, especially, clustering. We also show the principal space-filling curves, their construction, and their applications. Thereafter, we describe the algorithm, its pseudocode, and a fictional example.

Finally, we describe how to carry out the ZOOM process that will be performed in the evaluation and result refinement phase. We also comment on the future astronomical applications of this work and especially the ones that we want to begin and complete in the next academic year. In the Conclusion section, the highlights of this work are explained.

Lastly, in the Appendix, we showcase the OARMAC, WDS, HIPPARCOS, and TYCHO catalogs that we are familiar with and to which we want to apply our algorithm.

Keywords: Astronomy; Clustering; Hilbert Space-Filling Curve (HSFC).

Agradecimientos

A todos mis ancestr@s, en especial a los que he llegado a conocer y querer: Servando, Ana, Miguel, Bertila, Carmén y José Benito. A mis prim@s y tí@s. Muy especialmente, a mí hermana, Lucía,
Lux aeterna luceat semper .

A Juanín y Alex, y a tod@os mis amig@s cercanos o lejanos, a todos os llevo dentro.

A todos los profesores y estudiantes que he conocido durante estos años en las diferentes facultades de matemáticas en las que he podido crecer como estudiante. Gracias a la USC, a la UGR y a la Uniwersytet Jagielloński. Destacaría a tantos que no puedo destacar a ninguno.

También, a Andres Fraga, y al equipo de Disquecool, por su ayuda en la elaboración de parte de las imágenes que constan en este trabajo.

Gracias al equipo del OARMA y, en especial, a José Ángel Docobo Durantez, por su fe en mí, por su ayuda y su orientación. Este trabajo existe gracias a usted, su apoyo y sabiduría.

¡Gracias!

Santiago de Compostela, Julio 2013.

Jorge Gómez Crespo

Introducción

En 1943, el Dr. D. Ramón María Aller funda en la Universidad de Santiago de Compostela, el observatorio astronómico al que actualmente da nombre y cuyas siglas son OARMA. Dentro de las múltiples investigaciones y actividades que a lo largo de los años ha desarrollado el observatorio cabe destacar, por su interés científico y su repercusión, el catálogo de estrellas dobles **OARMAC**, promovido por el profesor J.A.Docobo [A.1], centrado en las órbitas y efemérides de este tipo de sistemas estelares. Actualmente alberga 1776 sistemas binarios y 2188 órbitas, siendo una referencia a nivel mundial. Cada binaria tiene asociada, además de sus siete elementos orbitales, diferentes tipos de parámetros físicos como los tipos espectrales, las magnitudes, la paralaje (inverso de la distancia), etc.

Como catálogo se centra en la descripción, lo más completa posible, de los objetos, en este caso estrellas binarias, que contiene. Pero, ¿qué información adicional podemos obtener? ¿Qué información se nos escapa a simple vista?. Y sí, en vez de 1776 objetos, fuesen cientos de miles, como en el caso del catálogo estelar Hipparcos [A.2], ¿podríamos obtener un conocimiento científico aprovechable de su estudio como conjunto?

El avance de las nuevas tecnologías, y en especial las encuadradas dentro del ámbito del KDD (Knowledge Discovery in Databases), ha generado un cambio de paradigma en nuestra forma de entender una base de datos. La búsqueda de patrones dentro de grandes bases de datos, a veces tan grandes que son imposibles de evaluar por procedimientos «manuales», ha adquirido en los últimos años una importancia enorme. Estos nuevos procedimientos y la velocidad con que evolucionan, sobre todo en temas relacionados con la telefonía móvil, o la geomática, suponen nuevos retos para los investigadores y científicos de todo el mundo. Estar al tanto de estas nuevas herramientas y adaptar las mismas a las áreas del saber clásicas, suponen un esfuerzo constante entre la comunidad científica mundial, muchas veces recompensado con nuevos descubrimientos o posibilidades.

Cabe destacar, tanto por la importancia que ha tomado en los últimos tiempos como por la gran variedad de campos donde se ha aplicado, el **clústering** (forma españolizada del término inglés *clustering*). Sus aplicaciones pueden encontrarse en campos como la genética, la biotecnología, las redes sociales, el marketing, ... y en general, en cualquier área donde una gran cantidad de datos pueda ser tamizada y organizada en grupos característicos o especialmente importantes. El «clústering» en sí, consiste en agrupar objetos basándonos en su similitud. Cuando hablamos del clústering espacial nos referimos a que esa similitud está marcada por una distancia, normalmente euclídea. En el presente trabajo queremos desarrollar un mecanismo de clústering propio, para ello, nos basaremos en trabajos de autores como Yun-Tai Lu [1], Chang-Tien Lu et al. [2] o Chua-Huang Huang et al. [3], principalmente.

Los conocimientos adquiridos en la asignatura de Astrodinámica, durante los estudios del Master en Matemáticas de la USC, permiten reconocer un creciente interés en en el desarrollo y mejora de grandes bases de datos astronómicos. Los resultados obtenidos en misiones como Hipparcos o Plank alcanzaron una amplia difusión internacional y suscitaron un gran interés por parte de la opinión pública. Muchísimos otros proyectos han surgido en las últimas décadas y algunos muy ambiciosos están actualmente en marcha, como por ejemplo, la misión Gaia [A.4] de la Agencia Espacial Europea (ESA).

Gracias a las lecturas desarrolladas en Biomatemáticas, considero la multidisciplinaridad una opción más a la hora de enfrentar un problema. Las herramientas matemáticas han ayudado a lo largo de la

historia a lograr avances determinantes tanto teóricos como sociales; las políticas de vacunación o el estudio del comportamiento dinámico de una enfermedad son ejemplos notables de ello. Otro ejemplo es la aplicación de la Teoría de Grafos al estudio del metabolismo o de las redes sociales. Nosotros recorremos en este trabajo el camino inverso. Buscamos aplicar otras ramas al saber astronómico. Los métodos del data mining nos parecen los idóneos para conseguir un aprovechamiento óptimo de las grandes bases de datos estelares, los **catálogos astronómicos**, [A]. Como estos son cada vez más grandes (el **Tycho** 2 [A.3] contiene datos de 2.539.913 estrellas), es conveniente aplicar el clústering espacial para la búsqueda de patrones o tendencias que puedan existir entre ciertas características de las estrellas catalogadas.

Jakub Schwarzmeier [5] siguiendo el algoritmo Barnes-Hut, tree-code o tree-code jerárquico utiliza la curva, desarrollada por D. Hilbert en el año 1891 [4], que llena el espacio para enfrentarse al problema de la simulación galáctica. Esta curva nos permite preservar cierta localidad y jerarquía a la hora de reducir el problema del estudio dinámico de N objetos al estudio de m celdas donde agrupamos los datos, con $m << N$. La **curva de Hilbert** y el clústering se asocian en muchos de los productos que la tecnología *Wireless* nos proporciona en nuestro día a día.

Quizás el ejemplo más sorprendente se produce en el campo de la robótica, y en especial, en la definida como tecnología Wireless Sensor Nodes (**WSNs**). La WSNs se usa principalmente para el estudio del medioambiente y el movimiento de robots. Consiste en un conjunto de diminutos sensores autónomos para monitear las condiciones físicas circundantes. Es decir, estudían, comunican y/o procesan información tras un proceso de recopilación de grandes cantidades de datos recogidos en su entorno sensible. Como es de imaginar, esta tecnología se aplica al estudio de la contaminación atmosférica, a la detección de incendios forestales y urbanos, a la monitorización tanto médica como de aguas residuales, e incluso al estudio de tornados. Para todos es conocida la figura del cazatornados y la de sus extraños aparatos que contienen miles de sensores. El objetivo es que sean tragados por el tornado y se desperdiguen por toda su estructura con el fin de mandar información sobre la temperatura, la intensidad, la dirección, ... desde dentro del propio vórtice. Los métodos de clústering en el procesamiento de datos procedentes de los sensores o *motes*, y la curva de Hilbert a la hora de recorrer cada sensor su área de monitoreo son fundamentales para todas estos sistemas. Por otra parte, su uso militar es bien conocido por todos. Los nuevos instrumentos de supervivencia en combate que portan los soldados, o los robots autónomos o semi-autónomos son realidades que desgraciadamente no nos son ajenas.

A nivel de calle, gran cantidad de servicios telemáticos comunes (e.g. servicios de tráfico, localización de restaurantes, programas de navegación en carretera, condiciones atmosféricas y de polución, etc.) son altamente dependientes de la localización del usuario y acceden a grandes bases de datos con el fin de suministrar información a dicho usuario de manera rápida y, a ser posible, con un bajo coste energético. Siguiendo estos principios se han desarrollado muchas estrategías de búsqueda. Las basadas en el índice de la curva de Hilbert (HCI) resultan muy útiles pues permiten compaginar las búsquedas en ventanas (*windows queries*) con las búsquedas en entornos cercanos. Podemos encontrar una buena introducción al tema en [6], y avances más recientes en [7], [8] y [9]. En todo caso, este no es el objetivo de nuestro trabajo.

Nosotros hemos buscado desarrollar un algoritmo eficiente que no sólo agrupe las celdas relativamente densas que están próximas, sino que también agrupe las vacías o poco densas. Como nuestra intención es crear un algoritmo dirigido al estudio de los catálogos estelares, hemos querido aprovechar el hecho de que el espacio es en gran parte vacío. De esta manera aparece una condición de parada natural, buscar que exista una sola componente conexa de vacío en nuestro clústering espacial, lo que viene a ser que, el clúster de vacío agrupe todas las celdas que no son lo suficientemente densas para ser consideradas como tal. Obviamente puede ocurrir que, después de hacer correr el algoritmo una vez, nos queden varias componentes, lo que nos lleva al concepto de hacer zoom. Con hacer zoom nos referimos a seleccionar una zona en particular del enrejado, y volver a aplicar el proceso en ella.

Lo más innovador de nuestro método, y lo que más dificultades técnicas ha presentado, ha sido que el algoritmo que se propone agrupa en el mismo clúster celdas contiguas similares, aunque estas estén situadas en diagonal, por lo que los saltos dentro de la curva de Hilbert se vuelven muy difíciles de evaluar por la mayoría de los métodos en uso.

Para solucionar esta problemática, hemos buscado una solución basada en un indexado interesado de curvas de Hilbert de orden menor que llenen los cuadrantes de otra curva base de orden mayor. En el método que proponemos hemos tomado una curva de Hilbert de orden 3 para que nos dote a toda la base de datos de una estructura de rejilla de 64 celdas, y de un indice u orden prefijado a la hora de recorrerlas. Por otro lado, hemos dividido la rejilla en cuatro cuadrantes y hemos aplicado una curva de Hilbert de orden 2, orientada de un modo particular, a cada uno de ellos. Este método es aplicable a ordenes mayores que 3 y 2, pero requiere un conocimiento muy detallado de las curvas con los órdenes a utilizar y un procedimiento muy puntilloso que se dificulta cuanto mayor sea el orden de la curva.

La potencia del método radica en su sencillez, tanto intelectual como computacional, y en el hecho de que siempre se puede volver a aplicar el mismo procedimiento a zonas de interés para pulir o mejorar los resultados, ZOOM. Si aplicamos una vez el procedimiento, el espacio de datos se ha reducido a 64 celdillas a evalúar pero su resultado puede ser demasiado grueso. Si esto ocurre, podemos seleccionar la zona , el conjunto de celdas, que deseemos y volver a aplicarle el procedimiento.

Supongamos que solamente se lo aplicamos a una celda. Si aplicamos el procedimiento una segunda vez a una sola celda, el tamaño de las celdas resultantes correspondería con el tamaño de celda de una partición global de orden 6, es decir, dividir el espacio de datos en $(2^3 \times 2^3)^2 = 2^6 \times 2^6$ celdas; si lo aplicamos tres veces a una sola celda, el tamaño correspondería al de las $(2^3 \times 2^3)^3$ celdillas correspondientes con el orden 9; etc. Si tomamos una zona con $m \times m$ celdillas, el dominio correspondiente a estas celdillas se partirá entre 64 celdas, y cada una de estas celdillas abarcará unas hipotéticas m^2 celdas de orden 6, correspondientes con haber aplicado la curva de orden 6 a todo el espacio de datos original.

En el desarrollo del trabajo introducimos, en la Secc. 1 una serie de conceptos básicos, tanto de *Data Mining* como de *Clustering*. En la Secc. 2 hablamos ligeramente de las curvas que llenan el espacio y en particular de las tres más conocidas, Peano, RBG y Hilbert. En la Secc. 3 describimos el algoritmo que queremos desarrollar. Comentaremos sus fundamentos, el tratamiento previo de los datos, una descripción de su funcionamiento paso a paso, el programa principal y los pseudocódigos de cada procedimiento realizado. En la secc. 4 explicaremos someramente el funcionamiento del procedimiento de post-evaluación ZOOM. En la secc. 5 esbozaremos las posibles aplicaciones en astronomía que pueden surgir de este trabajo. En la secc. 6 se da cuenta de algunos de los futuros trabajos que se plantean desarrollar el año académico que viene. Y finalmente, en el apéndice A describimos someramente los principales catálogos astronómicos que hemos ido nombrando a lo largo del trabajo.

Motivación.

Las herramientas de análisis que han surgido en los últimos años de la mano de la geomática y los recientísimos desarrollos en tecnología Wireless, especialmente en los problemas de localización y búsqueda en entornos cercanos; nos han motivado a investigar sus posibles aplicaciones al campo de los catálogos estelares. Los catálogos estelares, como el **Hipparcos** [A.3] o el **Washington Double Star** (**WDS**) [A.2], contienen una grandísima cantidad de información y creemos que es posible ampliar sus posibilidades siguiendo las pautas que la KDD (*Descubrimiento de Konocimiento en Bases de Datos*) ha aplicado con éxito en otros campos.

Dentro de esta motivación general hemos buscamos adaptar algunas herramientas del clústering al campo de la información astrofísica. En particular, presentamos un algoritmo de clustering para grandes bases de datos, que realiza un clústering de vacío o ruído, y otro de zonas densas. Es nuestra intención que este trabajo muestre el potencial de estas aplicaciones para aprovechar en mayor medida el conocimiento que reúnen. Otras iniciativas se están llevando a cabo en el ámbito nacional como el proyecto de la UCM dirigida por Rafael Caballero Roldán, [10]; a nivel internacional, es un buen ejemplo, el trabajo de Nicholas M. Ball del Herzberg Institute of Astrophysics en Canada y Robert J. Brunner de la Universidad de Illinois, [11].

Objetivos.

Crear un algoritmo para el caso 2-dimensional que por un lado efectúe una búsqueda y conglomerado eficiente de los posibles clústers desde el punto de vista clásico de localización espacial con restricciones físicas, y en particular astrofísicas. Por un lado, agruparemos los clústeres densos y por otro trataremos de agrupar el vacío, o ruido, en la menor cantidad posible de componentes conexas. Consideraremos que dos celdas son cercanas si son adyacentes, incluso en el caso de que solo compartan un punto en común, i.e. diagonalmente.

Capítulo 1

Conceptos básicos

El *data mining*, o minería de datos (**DM**), es la etapa de análisis del KDD (*Knowledge Discovery in Databases*); su fin es obtener patrones dentro de las grandes bases de datos que en los últimos años se han recopilado en una gran variedad de campos como la genética, la ingienería eléctrica, los hábitos de compra, los recursos humanos, ... Un proceso típico de minería de datos consiste en el siguiente protocolo:

1. Selección del conjunto de datos
2. Análisis de las propiedades de los datos
3. Transformación o preprocesamiento del conjunto de datos de entrada
4. Seleccionar y aplicar la técnica de minería de datos
5. Extracción de conocimiento
6. Interpretación y evaluación de datos

Las técnicas del DM proceden en gran medida de la Inteligencia Artificial y de la Estadística, y vienen a consistir en algoritmos que se aplican sobre un conjunto de datos para obtener unos resultados más o menos predecibles. Se clasifican estos algoritmos en dos grandes tipos: *algoritmos supervisados*, o *predictivos*; y *algoritmos no supervisados*, o *del descubrimiento del conocimiento*. Los primeros predicen un dato o un conjunto de los mismo a partir de otros conocidos; mientras que los segundos, buscan descubrir patrones o tendencias entre ellos. Una de las técnicas más representativas, normalmente incluída dentro de los algoritmos de aprendizaje no supervisado, es el agrupamiento o *clustering*, donde se produce una partición del conjunto de partida en subconjuntos o *clusters*.

El clústering, por tanto, consiste en un algoritmo que agrupa una serie de datos según criterios habitualmente de distancia, aunque se puede adaptar a otros tipos de parámetros cualitativos, en función de su cercanía, o similitud. Para ello se define la cercanía en función de una distancia, euclídea generalmente, o una propiedad estadística llamada verosimilitud. Se busca que los datos que acaben perteneciendo a un mismo clúster compartan propiedades comunes. El conocimiento de estos grupos puede permitir una descripción sintética de un conjunto de datos multidimensional complejo. Esta descripción se consigue sustituyendo la descripción de todos los elementos de un clúster por la de un representante característico del mismo. De ahí su uso en diferentes ramas como la biología, la medicina, el marketing, la teoría de la señal, la biometría o, en el caso particular de este trabajo, la astronomía.

Se pueden dividir los algoritmos de clustering en cuatro grandes bloques:

1. **De partición**: construye una partición de la base de datos **D** formada por N objetos n-dimensionales en k clústers, siendo este un valor que hay que introducir manualmente. Cada clúster se representa por el centro de gravedad del clúster (*k-means method*) o por uno de los objetos localizado cerca de dicho centro (*k-medois method*)[12]. Los ejemplos más conocidos de este tipo de clustering son: *PAM*[13], *CLARA*[13] y *CLARANS*[14], además del propio *k-means*[13].

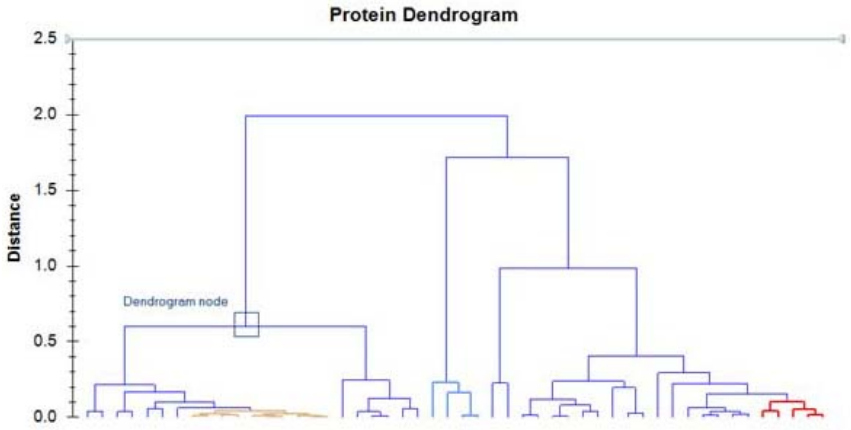

Figura 1.1: Ejemplo de dendrograma

2. **Jerárquicos**: El *clústering jerárquico*, o basado en la conectividad, parte de agrupar objetos cercanos, o parecidos basándose para ello en cierta «distancia» prefijada, o función de similitud. De esta forma conectan «objetos» hasta formar un clúster basándose en esa similitud seleccionada. Se formaran diferentes particiones, que se pueden representar usando un dendrograma (ver Figura 1.1), un árbol que iterativamente subdivide el conjunto de datos, **D**, en subconjuntos más pequeños hasta que cada subconjunto consiste en un solo objeto, salvo que incluyamos *condiciones de parada* que se verifiquen antes de que eso pase. Es decir, no proporcionan una sola partición del conjunto de datos, más bien, nos permiten organizar jerárquicamente diferentes capas de clústers que surgen en cada uno de los pasos del algoritmo. En un dendrograma, el eje de las y nos muestra los diferentes clústers que surgen en el paso correspondiente, mientras que los diferentes grupos que aparecen a una misma altura a lo largo del eje de las x son grupos diferenciados a un mismo nivel. En resumen, crean una descomposición jerárquica de la base de datos D.

   Estos métodos se subdividen en:

   a) *Aglomerativos*: cada objeto empieza siendo en sí mismo un cluster, estos se van aunando formando nuevos clusters con más de un elemento hasta llegar a formar el clúster que abarca a todos los objetos; en caso de que algún objeto no se llegase a juntar con ningún otro permanecería inalterable como clúster en sí mismo a lo largo de todo el proceso hasta el último paso donde se junta para formar el conjunto total.

   b) *Divisivos*: se empieza con un sólo clúster que abarca a todos los objetos y se va subdividiendo progresivamente en otros hasta que cada elemento forma un clúster en sí mismo. La condición de parada más común es que se forme un predeterminado número de clústers.

   En general, la complejidad del clustering aglomerativo es del orden de $O(n^3)$ lo que lo hace demasiado lento para grandes bases de datos. En el caso del clustering divisivo, su complejidad es del tipo $O(2^n)$, es decir, incluso peor que los algoritmos del subcaso anterior. Los algoritmos más comunes de esta clase son: *HAC*[15], *BIRCH*[16], *ROCK*[17], *CURE*[18] y *CHAMELEON*[19].

3. **Basados en la densidad o *density-based***: aplica un criterio de clústering local. Los clústers son vistos como regiones en la base de datos dentro de los cuales los objetos se interpretan, a su vez, como regiones densas, y que están separados por regiones de baja densidad (ruido). La idea principal que subyace en estos métodos es que para cada objeto de un clúster, su entorno, fijado cierto radio (ϵ), tiene que contener un número mínimo de objetos (*MinPts*), es decir, el número de objetos debe superar un determinado umbral dentro de un cierto entorno del objeto.

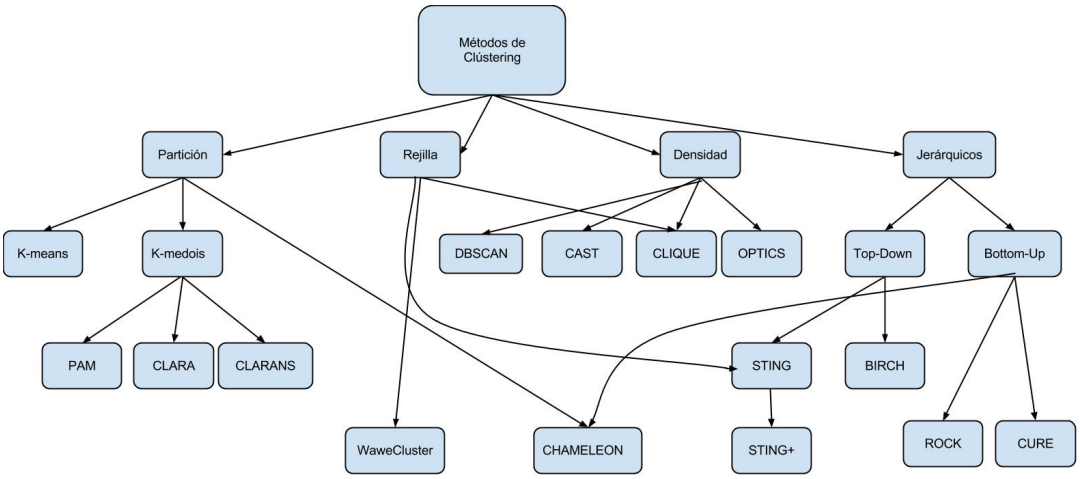

Figura 1.2: Diagrama de los principales metodos de clustering

Son algoritmos basados en la densidad: *CLIQUE*[20], *OPTICS*[21], *CAST*[22], *DBSCAN*[12] y *WaveCluster*[23]

4. **Basados en una rejilla o *grid-based methods***: dividen el espacio de datos en un número finito de celdas y realizan el clustering sobre ellas. Es decir, se centran principalmente en la partición creada del espacio en vez de centrarse en los propios datos que lo conforman. La partición de los datos es inducida por su pertenencia a los segmentos o subespacios resultantes de la partición del espacio de datos. Una de sus ventajas más salientables es que en este proceso indirecto (*Conjunto de datos* ⇒ *grid-data* ⇒ *partición del espacio* ⇒ *partición de los datos*) la acumulación de grid-data permite que sean independientes de la ordenación de los propios datos, mientras que los métodos de reubicación y todos los métodos incrementales dependen mucho de ella.

   Su principal ventaja es que la velocidad del algoritmo sólo depende de la resolución del enrejado, pero no del tamaño del espacio de datos. Son por tanto los métodos a emplear en el caso de tener una alta densidad de objetos.

   Algunos algoritmos de esta clase son: *CLIQUE*[20], *STING*[24] y *WaveCluster*[23].

Capítulo 2

Curvas que llenan el espacio

El rápido crecimiento de las bases de datos, tanto cuantitativamente como cualitativamente, hace que las herramientas de clústering vayan tomando cada vez más relevancia, al igual que el resto de procesos propios del *data minig*. Como ya hemos comentado la aproximación basada en rejillas es muy eficiente cuando nos enfrentamos a grandes bases de datos, y en especial si tenemos una gran cantidad de datos por celda. Su principal ventaja es la de olvidarse de los datos en sí y centrarse en una cantidad mucho menor de objetos a analizar, las celdas o bloques que delimita el enrejado.

Una curva que llena el espacio (*Space-Filling Curve*), de ahora en adelante **SFC**'s, es un camino que pasa por cada punto del espacio una sola vez de forma que construye una correspondencia entre las coordenadas de los puntos y una secuencia unidimensional de números, índice, asignados a los puntos de la curva. Nos proporciona un sistema para ordenar linealmente las celdas de una rejilla. Como sabemos, al ser al espacio de datos un espacio 2-dimensional o superior, no existe un orden total que preserve la proximidad espacial. Quiere esto decir, no existe un mapa que lleve datos con dos o más dimensiones a una dimensión y que, a su vez, verifique que cualquier par de objetos cercanos en el espacio d-dimensional en cuestión sigan siendo cercanos para cualquier orden que le apliquemos al índice 1-dimensional.

La principal característica de las SFC's es que preservan las distancias, es decir, dos celdas con índices lineales consecutivos van a estar necesariamente contiguas en el espacio de datos y por tanto representar objetos similares. Pero no necesariamente dos celdas han de ser consecutivas para que sus datos estén relacionados por similitud. A pesar de estas deficiencias, su utilidad radica en que nos permiten preservar parte de la localidad espacial y aumentar la velocidad en el cálculo de los clusters.

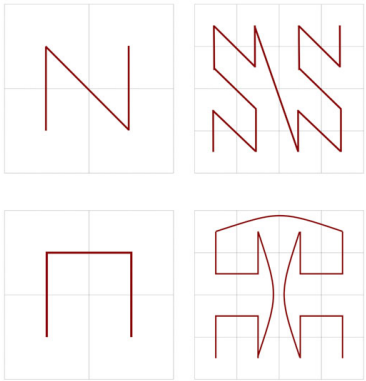

Figura 2.1: Curvas de Peano y RGB de orden 1 y 2

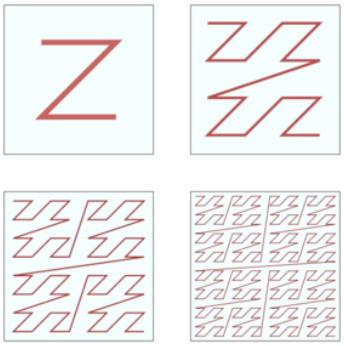

Figura 2.2: Curvas de Peano de orden 1, 2, 3 y 4

En general, las SFC's empiezan con un camino básico en una rejilla cuádrica k-dimensional. El camino visita cada celda una vez sin cortarse a sí misma. Tiene dos finales libres a los que podemos unirles nuevos caminos. La curva base se dice de orden 1. De forma iterativa podemos llegar a una curva de orden h sustituyendo cada vértice de la curva básica por una curva de orden *h-1*, rotada y/o reflejada adecuadamente.

Cronológicamente, la primera SFC fue la **curva de Peano** [25], ideada en 1890. Jack A. Orenstein introduce el concepto dentro del KDD en el año 1984 {[26] [27]}. En este mapa la secuencia 1-dimensional de números (1D-number) asociada a los puntos se obtiene por un simple intercalado de los bits de la representación binaria de las coordenadas x e y de los puntos de un espacio 2-dimensional (**2D-espacio**). La pieza básica de la curva de Peano para una rejilla 2×2, que denotamos P_1 se muestra, arriba en la Figura 2.1. Para llegar a una curva de esta clase de cualquier orden superior, reemplazamos cada vértice de esta curva básica por la curva del orden anterior. En la Figura 2.2 se nos muestran las curvas de hasta orden 4. Podemos pensarlo como: dividimos el espacio de datos 2-dimensional en cuadrantes, dibujamos la curva de Peano que sirve para numerarlos. Después dividimos cada cuadrante en cuatro nuevos subcuadrantes, y la misma curva básica de orden 1, manteniendo el sentido de la numeración, se dibuja en cada uno de ellos, reemplazando el vértice de la curva anterior y uniendo el final de cada cuadrante con el origen del siguiente (final y origen en cuanto a la numeración de la curva básica). Y así, iterativamente, hasta llegar al orden deseado.

Recordemos que, usando el código de Gray [28], los números se codifican en una representación binaria tal que números sucesivos difieren solamente por la posición de uno de sus bits. Faloutsos {[29], [30]} comenta que esta diferencia en un solo bit está relacionada con la localidad. Propone que los números producidos mediante el intercalado de las coordenadas de un punto 2-dimensional, como el realizado por la curva de Peano, nos permite obtener un 1D-número, o **índice lineal**. La pieza básica de la **curva RBG**, *reflected binary Gray-code curve* de una rejilla 2×2 se muestra en la Figura 2.1 abajo a la izquierda, y se suele denotar como R_1, el procedimiento para obtener de forma iterativa ordenes superiores de la curva consiste en reflejar la curva de orden anterior sobre el eje de las x's y posteriormente sobre el eje de las y's.

La **curva de Hilbert** {[4], [31], [32]}, o **CH**, es una aplicación en la que los cuatro cuadrantes de un 2D-espacio se llevan a puntos cercanos en la transversa. Empieza con la primera pieza mostrada en la Figura 2.3, las curvas hasta orden seis se muestran sucesivamente en la misma figura. La pieza básica de la curva de Hilbert se denota por H_1, donde el vértice cero es el de abajo a la izquierda y el vértice 3 el de abajo a la derecha. Al igual que en las curvas anteriores, se replica en los cuatro cuadrantes. En el replicado, en el cuadrante de abajo a la izquierda, la pieza base, o baldosa fundamental, sufre una rotación de 90°en el sentido de las agujas del reloj y una reflexión respecto a la transversal; en el cuadrante de abajo a la derecha se rota en sentido antihorario otros 90°y también sufre una reflexión; es decir, tras la reflexión el sentido, o dirección transversal, de ambos cuadrantes se invierte. Los cuadrantes superiores, no sufren rotación ni inversión de la numeración, o en la dirección transversal. Para subir el orden de la curva el procedimiento que se sigue es el mismo, rotar y reflejar la curva de orden justamente inferior en los vértices 0 y 3, al igual que de H_1 a H_2.

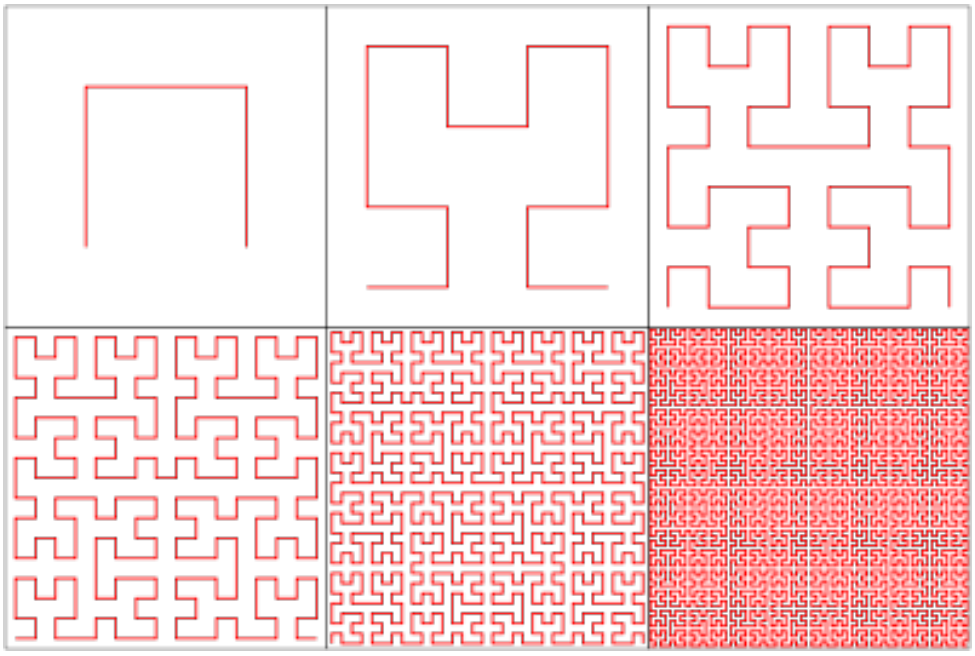

Figura 2.3: Curvas de Hilbert hasta orden 6

Para un mapa que conserve las distancias y que lleve los puntos 2D-dimensionales en un índice, 1-dimensional, un clúster se define por ser el grupo de objetos, o puntos, con valores consecutivos en el índice. Si el rango de búsqueda devuelve pocos clústers, los métodos basados en aplicaciones que conservan la distancia requieren poco acceso a la memoria, y por tanto son más rápidos.

Capítulo 3

Algoritmo de Clustering

3.1. Fundamentos.

Vamos a combinar la estrategia jerárquica dada por la SFC de Hilbert (es ampliamente aceptado que los métodos que usan este tipo de curva, que llamaremos **HSFC**, produce el mejor clustering [17]), con la estructura que proporciona el método grid-based o de rejilla. Este último método sirve para simplificar el problema del estudio de N-objetos y centrar nuestra atención en las m celdas que constituyan el enrejado, en las que se agrupan los datos por pertenencia con un solo acceso a los mismos. Claramente, nos interesa que $m << N$.

Usaremos una rejilla numerada por la HSFC para proveer de un orden lineal a las celdas de la rejilla. Su punto fuerte es mantener la distancia de aquellos objetos que son vecinos en el espacio 2D pero pertenecen a celdas diferentes. Este tipo de métodos se ha mostrado eficiente en otras ocasiones sobre todo cuando el número de objetos aumenta.

Mientras que los algoritmos del tipo k-means elevan considerablemente su tiempo de ejecución al aumentar el número de datos a evaluar, los algoritmos que mezclan la HSFC con la rejilla asociada al orden de la misma muestran un comportamiento más adecuado, con un incremento de los tiempos mucho más suave. Además los k-means muestran una gran dependencia respecto a k en cuanto a la calidad del clustering. En el tipo de algoritmos que usamos en este trabajo existe una dependencia con el orden de HSFC h, pero no muy acusada.

Las estrategias típicas para mejorar el rendimiento de los algoritmos de clústering basados en las HSFC's usan procesos de «shift» o multiíndices. Lo bueno de nuestro planteamiento es su sencillez, con sólo dos índices y unas cuantas comprobaciones podemos asegurarnos que todas las celdas contiguas sean evaluadas como mínimo una vez y que se reenumeren los números de cluster denso y vacío las veces necesarias para que puedan aunar todos los clústers posibles. En caso de necesitar más precisión siempre es posible dividir los clusters densos en nuevas subceldillas y volver a iterar el algoritmo. A esta técnica la llamaremos ZOOM

3.2. Tratamiento de datos.

Antes de hacer correr el algoritmo de clustering debemos asignar cada objeto incluído en la base de datos a una de las celdas de la rejilla. El orden h de la HSFC seleccionada, indica que el número total de celdas es $2^h \times 2^h = m$. Es decir, dividiremos el espacio base en m celdas, resultantes de la división en cuatro cuadrantes de cada una de las celdas provenientes del orden anterior, y se mirará cuantos objetos pertenecen a cada bloque. Orenstein {[11],[12]} usó el término **h-ordering** para referirse a la numeración de la HSFC de orden h; y a la curva en sí se le llama $H^{\mathbf{h}}$.

En nuestro método, las celdas serán numeradas por una HSFC de orden 3, del 0 al 63, el origen se situará en la celda inferior izquierda y su final en la celda inferior derecha. Fijémonos en que HSFC de

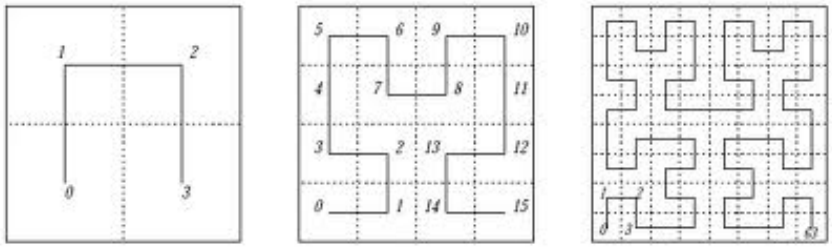

Figura 3.1: En esta figura podemos observar las numeraciones asociadas a los tres primeros órdenes de la curva de Hilbert. Los números dentro de cada celda representan el h-ordering.

orden 3 consta de cuatro piezas de orden dos. Las de los cuadrantes superiores sin rotación ni reflexión. La del último cuadrante rotada 90° en sentido antihorario y reflejada, y la del primero rotada 90° en sentido horario y reflejada, de forma que el origen coincide con la celda inferior izquierda (ver Figura 3.1 derecha).

Por otro lado, dividiremos la rejilla inicial de 64 celdillas, en cuatro partes, inferior izquierda, inferior derecha superior izquierda y superior derecha. Llamaremos, **P0**, a la esquina inferior izquierda, que en el 3-ordering corresponde a las celdas 0-15. **P1** la esquina inferior derecha, 48-63. **P3**, la esquina superior izquierda, celdas 16-31. **P2** la esquina superior derecha, 32-47.(Ver Figura 3.2)

Cada una de estas divisiones de 16 celdas se puede recorrer con una HSFC de orden 2. Vamos a movernos sobre P1 y P2 con las curvas de Hilbert de orden 2 de forma que los orígenes de cada una coincidan en la esquina inferior izquierda y derecha respectivamente (celdas 0 y 63 en la numeración de orden 3). Sus finales estarán enfrentados respecto al eje vertical central (celdas 5 y 58); y las superiores de forma que los orígenes coincidan también en las esquinas, celdas 21 y 42; y los finales enfrentados en el eje, 26 y 37.

Al sitúar los orígenes y finales de las curvas de esta manera conseguimos que las fronteras de los cuadrantes adyacentes compartan numeración dos a dos, simplificando la evaluación de los casos posibles. De esta manera, la frontera entre P0 y P1, y la frontera entre P2 y P3, tienen enfrentadas sus respectivas celdas 10, 11, 12 y 15. Mientras que las fronteras entre P0 y P3, y la que existe entre P1 y P2, tienen enfrentadas las celdas 5, 6, 9 y 10.

Al asociar cada objeto a una de las celdas, conseguimos reducir la complejidad de los cálculos. Cada celda vendrá numerada dos veces, una por la HSFC de orden 3 y la otra por el cuadrante y la HSFC de orden 2. Obviamente, existe una correspondencia biunívoca entre las celdas en la numeración de Hilbert orden 3 y la numeración en la curva de orden 2 según el cuadrante al que pertenezca, pero para simplificar esta relación utilizaremos un enrejado inicial, con numeración matricial que servirá de referencia para ambas numeraciones.

Una vez situado cada objeto en el bloque numerado que le corresponde en el enrejado numerado matricialmente, la HSFC de orden 3 nos proporciona un índice global; si dividimos el enrejado en las cuatro partes que comentábamos anteriormente (P0, P1, P2 y P3), la curva de orden 2 nos da un índice por cuadrante. Además cuantificaremos el número de objetos que posee cada bloque del enrejado y procederemos a la búsqueda de los clústers.

3.3. Algoritmo de procesamiento de datos.

- **Paso 1**: partimos de una rejilla de 64 celdas, con numeración matricial donde ubicamos los datos y los contabilizamos, de esta manera obtendremos un vector BI[C(ij]) con $i, j = 1, ..., 8$; que nos indicará cuantos objetos se encuentran ubicados en la celda ij, **C(i, j)**. Si $BI[C(i,j)] > \epsilon(h)$ la celda se considera **dense** o densa; en caso contrario, se considera **void** o vacía. Con esta evaluación obtendremos un vector **DV[C(i, j)]**, con valores 1 en las celdas $C(i,j)$ densas y 0 en las vacías.

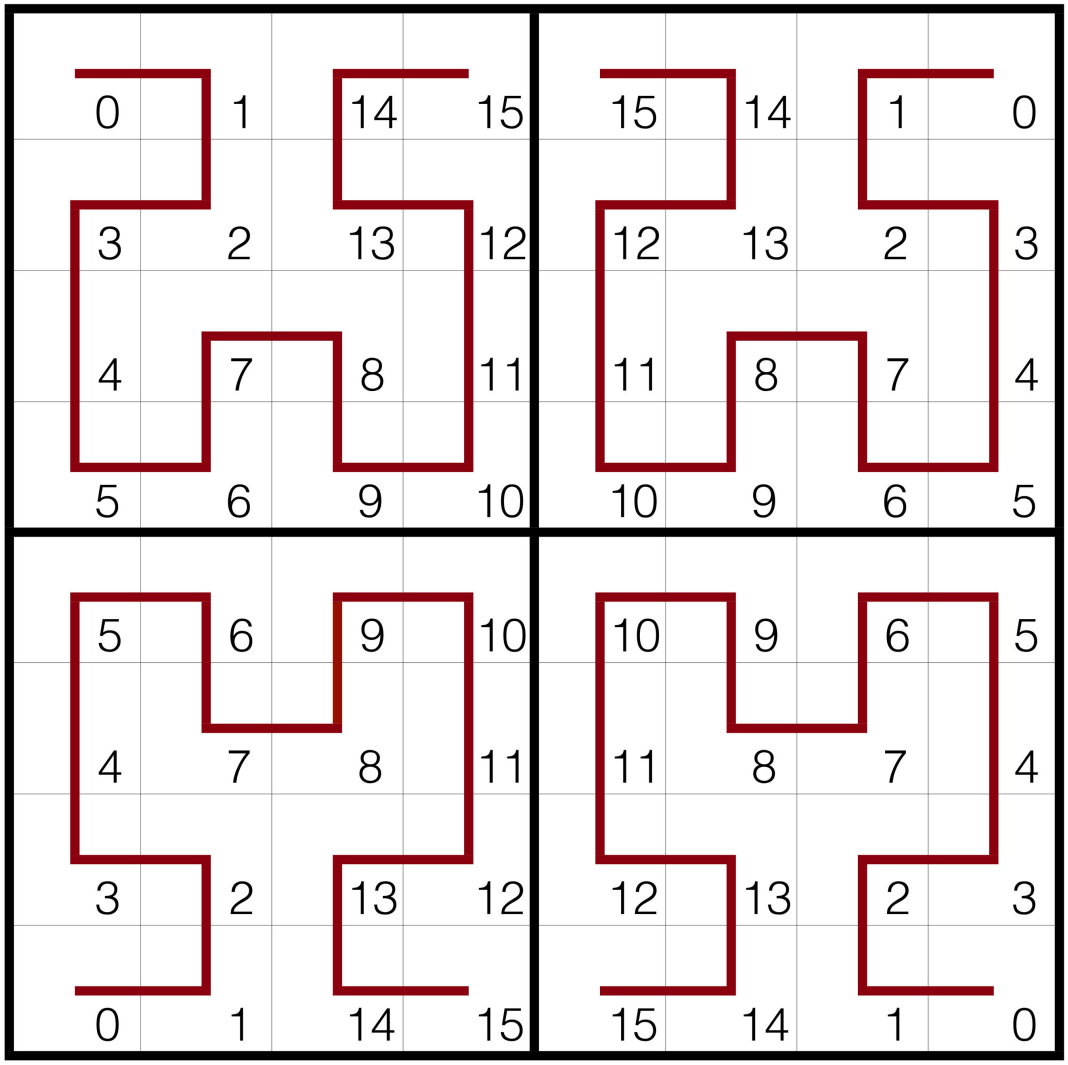

Figura 3.2: Dividimos en cuadrantes el enrejado inicial y trazamos las curvas de orden 2 con los orígenes en las esquinas.

- **Paso 2**: trabajaremos ahora sobre el 3-ordering que nos proporciona la curva H^3. El índice 0 de la curva de Hilbert de orden tres corresponde a la celda C(1,1), y el índice 63 a la celda C(1,8). Si a la celda i en la curva de Hilbert le corresponde la celda $C(j,k)$ del enrejado matricial, el vector **DV3[i]=DV[C(j,k)]** asocia a la numeración del 3-ordering el estado denso o vacío que le corresponde a cada celdilla.

- **Paso 3**: vamos a realizar otra organización en cuadrantes. Tal como se comentó anteriormente, el cuadrante **P0** incluye las celdas C(i,j) con $i = 1, .., 4$ y $j = 1, .,4$, es decir, el cuadrante inferior izquierdo. El cuadrante **P1** será el inferior derecho. **P3** será el superior izquierdo y **P2** el superior derecho. Son dieciséis celdas por cuadrante.

- **Paso 4**: a cada celda C(i,j), además del índice de H^3, le vamos a asociar otro índice según el cuadrante al que pertenezca. Para ello usaremos curvas de Hilbert de orden 2, con los orígenes en las esquinas y los finales enfrentados, como comentamos en el apartado anterior.

3.4. Algoritmo de clústering.

- **Paso 1**: buscamos identificar los bloques con densidad suficiente para ser considerados «densos» e ir aglomerando los mismos en clústers numerados. Al mismo tiempo hacemos lo mismo con aquellos que no llegan a la densidad suficiente y que consideramos «vacíos». Asociaremos las celdas contiguas que sean del mismo tipo, es decir, ambas **dense** o ambas **void**. Para enumerar cada clúster, denso o vacío, usaremos los indicadores $BCD3$ y $BCV3$, respectivamente.

  El procedimiento que denominamos **Lineal3OEvaluation(DV3)** analiza linealmente cada una de las 64 celdas en las que hemos dividido los datos siguiendo la numeración de H^3, a partir del vector **DV3[i]**. Recordemos que DV3[i] = 0 si la celda i es vacía, mientras que DV3[i] = 1 si es «densa».

  Si la celda i es densa, DV3[i]=1, la primera parte del algoritmo le asociará el número de clúster denso que le corresponda, **BCD3[i]** $= k$, y numerará los densos contiguos linealmente. Si la celda i es vacía BCD3[i]=0. De la misma manera se irán numerando los clústers vacíos, usando **BCV3[i]** $= j$, es decir, cuando $BI[C(ij)] \leqslant \epsilon(h)$ a la celda jk en la numeración de H^3 se le asocia el valor BCV3[i] que le corresponda. Cuando una celda sea densa se le asociará en BCV3 el valor cero. Al pasar de un clúster vacío a uno lleno se aumenta el contador de los vacíos $j = j + 1$, reciprocamente, al pasar de uno denso a uno vacío se aumenta el de los densos, $k = k + 1$.

  Aún así los saltos en la numeración de la curva de Hilbert no nos garantiza el haber numerado de la misma forma todas las celdas contiguas que compartan estado, es decir, que ambas sean densas o vacías.

- **Paso 2**: asociamos a BCD[C(ij)] el BCD3[i] que le corresponda, y lo mismo haremos con BCV[C(ij)] asociándole un valor de BCV3[i]. Ahora tenemos para cada celda C(ij) del enrejado inicial, un valor de cluster denso, BCD[C(ij)] y otro de cluster vacío, BCV[C(ij)]. Pasaremos estos valores a los vectores **BCDP[J]** y **BCVP[J]**, con $J = 0, 1, 2, 3$, según le corresponda a cada celda C(i,j).

  Los vectores BCDP[J] y BCVP[J] guardan la númeración de los cústers en el índice de la CH de orden 2 correspondiente al cuadrante J. Se puede calcular facilmente, dividimos en cuatro los vectores BCD3[i] y BCV3[i], cada parte se corresponde con uno de los cuadrantes P0, P1, P2 ó P3. Como a cada celdilla numerada por el 3-ordering le corresponde una C(ij) y a esta un índice en el

2-ordering, podemos reenumerar los vectores siguiendo el 2-ordering. A estos nuevos vectores, resultantes de la partición y la reenumeración, les vamos a llamar **BCDP[J]** y **BCVP[J]**, con J=0,..3.

- **Paso 3**: usaremos el 2-ordering del cuadrante P0. El procedimiento **LinealDiagonal2OEvaluation (J,BCVP[J], BCDP[J])**, con J=0, además de agrupar sus celdas linealmente, consigue a agrupar cuasi-diagonalmente, es decir, las celdas i e $i+2$, en el 2-ordering, que sean del mismo tipo, ambas densas o vacías, y siempre que $i \neq 3, 10$. Además, evaluamos las celdas 0, 4, 8 y 12 con las celdas 3, 7, 11 y 15 respectivamente, al ser origen y final de una CH de orden 1 y por tanto estar en contacto.

  Para que no quede ningún salto por calcular dentro del cuadrante P0, previamente a la evaluación lineal y cuasi-diagonal, introduciremos bajo ciertos supuestos los valores de clústering de las celdas $1, 2, 3$ y 8 en las celdas $7, 8, 13$ y 14. Nos referiremos a este proceso de ahora en adelante y en el contexto adecuado como **extender** los valores de clústering por un cuadrante.

  Como pueden quedar zonas mal numeradas debido al orden de los saltos debemos aplicar a continuación **InverseLinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])** con J=0; con esto los coeficientes de clústering han sido extendidos por contacto en todo el cuadrante.

- **Paso 4**: con **CuadrantJumps(J,I,DF[J],VF[J],DF[I], VF[I])** evaluaremos los saltos entre cuadrantes siguiendo el orden $P0 \to P1$, $P0 \to P2$ y $P0 \to P3$. De esta manera introduciremos los números de clústering de P0 en el resto de cuadrantes a través de su frontera.

- **Paso 5**: extendemos los valores de clústering introducidos anteriormente en los cuadrantes P1 y P3, para ellos usamos **InverseLinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])** con J=1 y J=3. Lo realizamos en el cuadrante P3 para que la numeración más baja se imponga lo mejor posible.

  Para que la extensión en P1 quede completa debemos aplicar, al igual que en P0, el otro algoritmo, **LinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])** con J=1.

- **Paso 6**: evaluamos los saltos $P1 \to P2$ y $P1 \to P3$. **CuadrantJumps(J,I,DF[J],VF[J],DF[I], VF[I])** con J=1 e I=2,3.

- **Paso 7**: extendemos en P2, usamos **LinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])** e **InverseLinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])**, con J=2.

- **Paso 8**: evaluamos el salto $P2 \to P3$, y extendemos en P3 mediante **InverseLinealDiagonal2OEvaluation (J,BCVP[J],BCDP[J])** y **LinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J])**, con J=3.

Con estas evaluaciones hemos conseguido introducir los números de clustering, vacío o denso, más pequeños posibles desde el cuadrante cero en los restantes, desde el cuadrante uno en el segundo y tercero, y finalmente, desde el cuadrante 2 en el 3. Tras cada salto cuadrante a cuadrante se han extendido los valores introducidos de forma que puedan llegar a marcar todas las celdillas del cuadrante en caso de ser necesario.

3.5. Pseudocódigo.

El programa principal toma los outputs del procesamiento de datos:

1. m: número de bloques.
2. $\epsilon(h)$: la densidad media de objetos por bloque.
3. $BI[C(ij)]$: cantidad de objetos en la celda C(ij) del enrejado referencia (con numeración matricial).
4. $BI3[i]$: cantidad de objetos en el bloque i, respecto a la curva de orden 3.
5. $BP0[j], BP1[j], BP2[j]$ y $BP3[j]$: cantidad de objetos en la celda j respecto al 2-ordering del cuadrante correspondiente.
6. $DV[(Cij)]$: estado denso o vacío de la celda ij en el enrejado matricial.
7. $DV3[i]$: estado denso o vacío de la celda i en el índice dado por la CH de orden 3.

Y llama, por orden, a las siguientes subrutinas:

1. *Procedure* $Lineal3OEvaluation(DV3)$.
   - Etiquetado inicial
2. *Procedure* $LinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$ &
   Procedure $InverseLinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$; con J = 0.
   - Extender los valores clústering en P0
3. *Procedure* $CuadrantJumps(J, I, DF[J], VF[J], DF[I], VF[I])$; con J = 0 e I = 1, 2 y 3.
   - Pasar los valores de clústering de la frontera de P0 a las fronteras de P1, P2 y P3
4. *Procedure* $InverseLinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$; con J = 1 y J=3 &
   Procedure $LinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$; con J = 1.
   - Extender los valores clústering en P1
   - Extender parcialmente los valores clústering en P3
5. *Procedure* $CuadrantJumps(J, I, DF[J], VF[J], DF[I], VF[I])$; con J = 1 e I = 2 y 3.
   - Pasar los valores de clústering de la frontera de P1 a las fronteras de P2 y P3
6. *Procedure* $LinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$ &
   Procedure $InverseLinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$; con J = 2.
   - Extender los valores clústering en P2

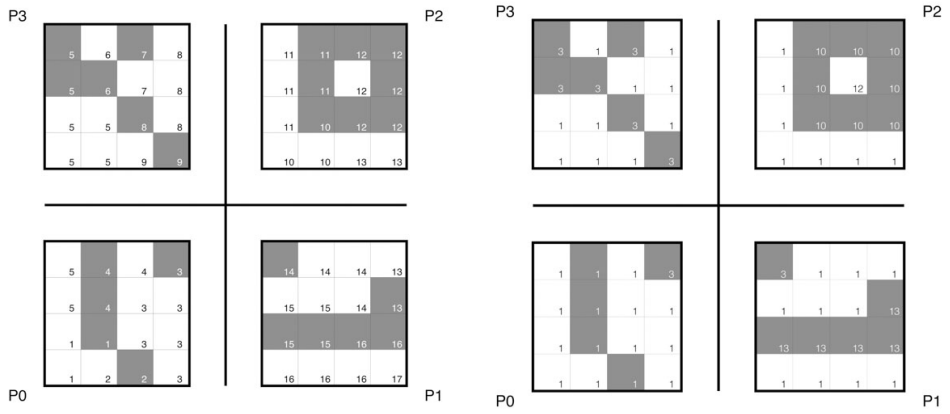

Figura 3.3: Etiquetado de clústers inicial y final de un espacio de datos enrejado en 64 celdillas.(Ver Ejemplo teórico, sección 3.6)

7. *Procedure* $CuadrantJumps(J, I, DF[J], VF[J], DF[I], VF[I])$; con J = 2 e I = 3.

   - Pasar los valores de clústering de la frontera de P2 a la frontera de P3

8. *Procedure* $InverseLinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$ &
   Procedure $LinealDiagonal2OEvaluation(J, BCVP[J], BCDP[J])$; con J = 3.

   - Extender los valores clústering en P3

Notemos que, entre el paso 1 y el paso 2 hay que transformar los output's BCV3[i] y BCD3[i], $i = 0, ..., 63$, con el etiquetado de clústers obtenido siguiendo H^3, a la numeración en la rejilla matricial, es decir, $BCV3[i] \rightarrow BCV[C(ij)]$ y $BCD3[i] \rightarrow BCD[C(ij)]$. Y, a continuación, pasar BCV[C(ij)] y BCD[C(ij)] a la numeración por cuadrantes, es decir, a los BCVP[J] y BCDP[J] con $J = 0, 1, 2$ *o* 3, según corresponda a su 2-ordering.

Al final del paso 8, si queremos recuperar la información en numeración matricial, tendremos que devolver los valores de cada BCVPJ[i] y BCDPJ[i] a las BCV[C(ij)] y BCD[C(ij)], según corresponda.

Como comentaremos en la introducción del pseudocódigo de **CuadrantJumps(J,I,DF[J],VF[J],DF[I], VF[I])**; tras efectuar los pasos 2, 4 y 6, hay que tomar los valores frontera de BCVP[J] y BCDP[J], BCVP[I] y BCDP[I] y modificarlos. Lo que haremos una vez fijados el par de I y J correspondientes, será seleccionar los valores BCVPJ[i], BCVPI[i], BCDPJ[i], BCDPI[i] con $i = 5, 6, 9, 10, 11, 12y15$; y los transformaremos en los inputs VFJ[j], VFI[j], DFJ[j] y DFI[j], con $j = 0, ..., 5$.

Estos VFJ, DFJ, VFI y DFI 's son los vectores de entrada de los pasos 3, 5 y 7. Tras efectuar dichos pasos, debemos volver a llevar los valores resultantes a cada posición correspondiente en BCVP[J] , BCDP[J], BCVP[I] y BCDP[I].

Los pseudocódigos de estos procedimientos siguen a continuación.

3.5.1. Lineal 3-Ordering Evaluation

El primer procedimiento, **Lineal3OEvaluation**, trabaja sobre el 3-ordering que nos proporciona la CH de orden 3 sobre el enrejado matricial. Este algoritmo nos da una numeración inicial de los clusters. Es una modificación sencilla del $ProcedureFirst$ en [1].

Procedure Lineal3OEvaluation(DV3)

1)$begin$
2) $j = 1$
3) $k = 1$
4) $Flag = 1$
5)$for\ i = 0\ to\ 63\ do$
6) $if(DV3[i] = 0)then$
7) $begin$
8) $BCV3[i] = j$
9) $BCD3[i] = 0$
10) $if(k > 1\ and\ Flag = 1)then$
11) $k = k + 1$
12) $elseif(k = 1)then$
13) $k = 1$
14) $endif$
15) $Flag = 0$
16) end
17) $else$
18) $begin$
19) $BCD3[i] = k$
19) $BCV3[i] = 0$
19) $if(Flag = 0)then$
20) $j = j + 1$
21) $endif$
22) $Flag = 1$
23) end
24) $endif$
25)end

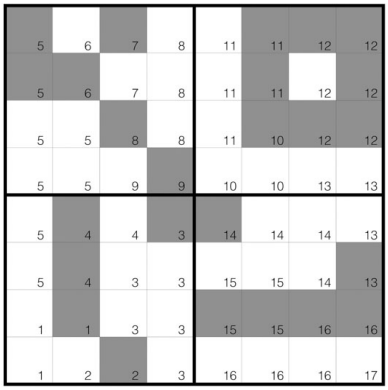

Figura 3.4: Etiquetado de clústers siguiendo el 3-ordering de un espacio de datos enrejado en 64 celdillas.(Ver Ejemplo teórico, sección 3.6)

3.5.2. Lineal-Diagonal 2-Ordering Evaluation

El segundo algoritmo trabaja sobre cada uno de los 2-orderings de cada cuadrante, evaluando linealmente y parcialmente en diagonal las adyacencias de cada celda, siempre que $i \neq 3, 10, 14$. En las celdas numeradas con 1, 2, 3 y 8 hacemos unos saltos para asegurarnos que sus índices entran en los otros tres cuadrantes de cuatro celdillas, de forma que se puedan extender al resto.

Procedure LinealDiagonal2OEvaluation(J,BCVP[J], BCDP[J])

1)*begin*
2)$for\ i = 0\ to\ 14\ do$
/***Evaluation 1, 2, 3, 8 ***/
3) $if(\ i = 1)\ \&\ (\ for\ j = 13, 14) then$
4) $if(\ i = 2)\ \&\ (\ for\ j = 7, 8, 13, 14) then$
5) $if(\ i = 3)\ \&\ (\ for\ j = 7) then$
6) $if(\ i = 8)\ \&\ (\ for\ j = 13) then$
7) *begin*
8) $if(BCVPJ[i] \neq BCVPJ[j]\ or\ BCDPJ[i] \neq BCDPJ[j]) then$
9) $if(BCVPJ[i] = BCVPJ[j] = 0\ \&\ BCDPJ[i] \neq BCDPJ[j]) then$
10) $if(BCDPJ[i] \leqslant BCDPJ[j])$
11) $BCDPJ[j] = BCDPJ[i]$
12) $if(BCDPJ[j] \leqslant BCDPJ[i])$
13) $BCDPJ[i] = BCDPJ[j]$
14) $endif$
15) $endif$
16) $if(BCDPJ[i] = BCDPJ[j] = 0\ \&\ BCVPJ[i] \neq BCVPJ[j]) then$
17) $if(BCVPJ[i] \leqslant BCVPJ[j)$
18) $BCVPJ[j] = BCVPJ[i]$
19) $if(BCDPJ[j] \leqslant BCDPJ[i])$
20) $BCVPJ[i] = BCVPJ[j]$
21) $endif$
22) $endif$
23) *end*
/***Evaluation Lineal saltos entre primeras y ultimas celdas curvas orden 1***/
24) $if(\ i = 0, 4, 8, 12) then$
25) *begin*
26) $if(BCVPJ[i] \neq BCVPJ[i+3]\ or\ BCDPJ[i] \neq BCDPJ[i+3]) then$
27) $if(BCVPJ[i] = BCVPJ[i+3] = 0\ \&\ BCDPJ[i] \neq BCDPJ[i+3]) then$
28) $if(BCDPJ[i] \leqslant BCDPJ[i+3])$
29) $BCDPJ[i+3] = BCDPJ[i]$
30) $if(BCDPJ[i+3] \leqslant BCDPJ[i])$
31) $BCDPJ[i] = BCDPJ[i+3]$
32) $endif$
33) $endif$

34) $if(BCDPJ[i] = BCDPJ[i+3] = 0$ & $BCVPJ[i] \neq BCVPJ[i+3])then$
35) $if(BCVPJ[i] \leqslant BCVPJ[i+3])$
36) $BCVPJ[i+3] = BCVPJ[i]$
37) $if(BCDPJ[i+3] \leqslant BCDPJ[i])$
38) $BCVPJ[i] = BCVPJ[i+3]$
39) $endif$
40) $endif$
41) end

/***Evaluation Lineal y diagonal salvo en 3 y 10***/

42) $if(BCDPJ[i] = BCDPJ[i+1] = 0)then$
43) $begin$
44) $if(BCVPJ[i+1] \leqslant BCVPJ[i])then$
45) $BCVPJ[i] = BCVPJ[i+1]$
46) $else$
47) $BCVPJ[i+1] = BCVPJ[i]$
48) $endif$
49) $if(i \neq 3, 10, 14)then$
50) $if(BCDPJ[i] = BCDPJ[i+2] = 0)then$
51) $if(BCVPJ[i+2] \leqslant BCVPJ[i])then$
52) $BCVPJ[i] = BCVPJ[i+2]$
53) $else$
54) $BCVPJ[i+2] = BCVPJ[i]$
55) $endif$
56) $endif$
57) $endif$
58) end
59) $endif$
60) $if(BCVPJ[i] = BCVPJ[i+1] = 0)then$
61) $begin$
62) $if(BCDPJ[i+1] \leqslant BCDPJ[i])then$
63) $BCDPJ[i] = BCDPJ[i+1]$
64) $else$
65) $BCDPJ[i+1] = BCDPJ[i]$
66) $endif$
67) $if(i \neq 3, 10, 14)then$
68) $if(BCVPJ[i] = BCVPJ[i+2] = 0)then$
69) $if(BCDPJ[i+2] \leqslant BCDPJ[i])then$
70) $BCDPJ[i] = BCDPJ[i+2]$
71) $else$
72) $BCDPJ[i+2] = BCDPJ[i]$
73) $endif$
74) $endif$
75) $endif$
76) end
77) $endif$
78) end

3.5.3. Inverse Lineal-Diagonal 2-Ordering Evaluation

Como en algunos casos vamos a necesitar realizar este procedimiento en sentido inverso definimos:

Procedure InverseLinealDiagonal2OEvaluation(J,BCVP[J],BCDP[J]),
como el mismo algoritmo pero con las líneas **2)** y **3)**, **4)**, **5)**, **6)**, **66)** y **84)**, modificadas por:

- **2) inverse)** $for\ i = 15\ to\ 1\ do$
- **3) inverse)** $if(i = 14)\ \&\ (for\ j = 1, 2) then$
- **4) inverse)** $if(i = 13)\ \&\ (for\ j = 1, 2, 7, 8) then$
- **5) inverse)** $if(i = 12)\ \&\ (for\ j = 8) then$
- **6) inverse)** $if(i = 7)\ \&\ (for\ j = 2) then$
- **24) inverse)** $if(i = 15, 11, 7, 3) then$
- **49) & 67) inverse)** $if(i \neq 1, 5, 12) then$

Consecuentemente, siempre que aparezca a lo largo del algoritmo **i+1**, **i+2** o **i+3**, pasaran a ser **i-1**, **i-2** y **i-3**, respectivamente. En la última parte del algoritmo, 42) en adelante, la labor del 3 la realiza el 5 y la del 10 el 12.

Procedure InverseLinealDiagonal2OEvaluation(J,BCVP[J], BCDP[J])

1) $begin$
2) $for\ i = 15\ to\ 1\ do$
/***Evaluation 7, 12, 13, 14 ***//
3) $if(\ i = 14)\ \&\ (\ for\ j = 1, 2) then$
4) $if(\ i = 13)\ \&\ (\ for\ j = 1, 2, 7, 8) then$
5) $if(\ i = 12)\ \&\ (\ for\ j = 8) then$
6) $if(\ i = 7)\ \&\ (\ for\ j = 2) then$
7) $begin$
8) $if(BCVPJ[i] \neq BCVPJ[j]\ or\ BCDPJ[i] \neq BCDPJ[j]) then$
9) $if(BCVPJ[i] = BCVPJ[j] = 0\ \&\ BCDPJ[i] \neq BCDPJ[j]) then$
10) $if(BCDPJ[i] \leqslant BCDPJ[j])$
11) $BCDPJ[j] = BCDPJ[i]$
12) $if(BCDPJ[j] \leqslant BCDPJ[i])$
13) $BCDPJ[i] = BCDPJ[j]$
14) $endif$
15) $endif$
16) $if(BCDPJ[i] = BCDPJ[j] = 0\ \&\ BCVPJ[i] \neq BCVPJ[j]) then$
17) $if(BCVPJ[i] \leqslant BCVPJ[j)$
18) $BCVPJ[j] = BCVPJ[i]$
19) $if(BCDPJ[j] \leqslant BCDPJ[i])$
20) $BCVPJ[i] = BCVPJ[j]$
21) $endif$
22) $endif$
23) end

/***Evaluation Lineal saltos entre primeras y ultimas celdas curvas orden 1***/

24) $if(\ i = 15, 11, 7, 3) then$
25) $begin$
26) $if(BCVPJ[i] \neq BCVPJ[i-3]\ or\ BCDPJ[i] \neq BCDPJ[i-3]) then$
27) $if(BCVPJ[i] = BCVPJ[i-3] = 0\ \&\ BCDPJ[i] \neq BCDPJ[i-3]) then$
28) $if(BCDPJ[i] \leqslant BCDPJ[i-3])$
29) $BCDPJ[i-3] = BCDPJ[i]$
30) $if(BCDPJ[i-3] \leqslant BCDPJ[i])$
31) $BCDPJ[i] = BCDPJ[i-3]$
32) $endif$
33) $endif$
34) $if(BCDPJ[i] = BCDPJ[i-3] = 0\ \&\ BCVPJ[i] \neq BCVPJ[i-3]) then$
35) $if(BCVPJ[i] \leqslant BCVPJ[i-3])$
36) $BCVPJ[i-3] = BCVPJ[i]$
37) $if(BCDPJ[i-3] \leqslant BCDPJ[i])$
38) $BCVPJ[i] = BCVPJ[i-3]$
39) $endif$
40) $endif$
41) end

/***Evaluation Lineal y diagonal salvo en 5 y 12***/

42) $if(BCDPJ[i] = BCDPJ[i-1] = 0) then$
43) $begin$
44) $if(BCVPJ[i-1] \leqslant BCVPJ[i]) then$
45) $BCVPJ[i] = BCVPJ[i-1]$
46) $else$
47) $BCVPJ[i-1] = BCVPJ[i]$
48) $endif$
49) $if(i \neq 12, 5, 1) then$
50) $if(BCDPJ[i] = BCDPJ[i-2] = 0) then$
51) $if(BCVPJ[i-2] \leqslant BCVPJ[i]) then$
52) $BCVPJ[i] = BCVPJ[i-2]$
53) $else$
54) $BCVPJ[i-2] = BCVPJ[i]$
55) $endif$
56) $endif$
57) $endif$
58) end
59) $endif$

60) $if(BCVPJ[i] = BCVPJ[i-1] = 0)then$
61) $\quad begin$
62) $\quad\quad if(BCDPJ[i-1] \leqslant BCDPJ[i])then$
63) $\quad\quad\quad BCDPJ[i] = BCDPJ[i-1]$
64) $\quad\quad else$
65) $\quad\quad\quad BCDPJ[i-1] = BCDPJ[i]$
66) $\quad\quad endif$
67) $\quad\quad if(i \neq 12, 5, 1)then$
68) $\quad\quad\quad if(BCVPJ[i] = BCVPJ[i-2] = 0)then$
69) $\quad\quad\quad\quad if(BCDPJ[i-2] \leqslant BCDPJ[i])then$
70) $\quad\quad\quad\quad\quad BCDPJ[i] = BCDPJ[i-2]$
71) $\quad\quad\quad\quad else$
72) $\quad\quad\quad\quad\quad BCDPJ[i-2] = BCDPJ[i]$
73) $\quad\quad\quad\quad endif$
74) $\quad\quad\quad endif$
75) $\quad\quad endif$
76) $\quad end$
77) $endif$
78)end

3.5.4. Cuadrant Jumps

Este procedimiento evalúa los saltos entre los cuadrantes. Hemos buscado, mediante la numeración dada por las curvas de orden 2, que las fronteras entre cuadrantes compartan dicha numeración, esto hace que la evaluación sea más sencilla. Para ganar en sencillez a la hora de escribir el algoritmo, reenumeramos la frontera. Las celdas 5, 6, 9, 10, 11, 12 y 15 en cada cuadrante ahora corresponderan a los índices 0, 1, 2, 3, 4, 5 y 6 respectivamente. De esta manera los nuevos índices 0,1,2,3 corresponden a las frontera entre P0 y P3, y entre P1 y P2. El resto corresponde a las celdas frontera de P0 y P1 y a las de P2 y P3. Además el índice 3 tiene que evaluar el salto de P0 a P2 y de P1 a P3.

También por sencillez, hemos llamado **DF0[i]** con $i = 0, ..., 6$ a BCDP0[j] con $j = 5, 6, 9, 10, 11, 12, 15$, con la relación entre i y j que comentábamos antes. De la misma manera VF0 se corresponde a BCVP0, DF1 con BCDP1, VF1 con BCVP1, ... La máquina no precisa de estas simplificaciones pero, a nosotros, nos ayudan a escribirlo más facilmente.

Según el valor de J e I, se tiene que hacer correr una parte u otra del algoritmo. Por ejemplo, si J= 0 e I= 2 actuaría solamente $CASE5$.

Procedure CuadrantJumps(J,I,DF[J],VF[J],DF[I], VF[I])

1)*begin*

– – – – –

/***CASE 1 J=0 and I=3 ***/**

/*** P0 → P3 ***/**

/***VOID***/**

2)$for\ i = 0\ to\ 3\ do$

3) $if(DF0[i] = DF3[i] = 0)then$

4) $VF3[i] = VF0[i]$

5) $elseif(i \neq 3\ and\ DF0[i] = DF3[i+1] = 0)then$

6) $VF3[i+1] = VF0[i]$

7) $elseif(i \neq 3\ and\ DF0[i+1] = DF3[i] = 0)then$

8) $VF3[i] = VF0[i+1]$

/***DENSE***/**

9) $elseif(VF0[i] = VF3[i] = 0)then$

10) $DF3[i] = DF0[i]$

11) $elseif(i \neq 3\ and\ VF0[i] = VF3[i+1] = 0)then$

12) $DF3[i+1] = DF0[i]$

13) $elseif(i \neq 3\ and\ VF0[i+1] = VF3[i] = 0)then$

14) $DF3[i] = DF0[i+1]$

15)*end*

– – – – –

– – – – –

/***CASE 2 J=1 and I=2 ***/**

/***P1 → P2 ***/**

/***VOID***/**

16)$for\ i = 0\ to\ 3\ do$

17) $if(DF1[i] = DF2[i] = 0)then$

18) $if(VF1[i] \leqslant VF2[i])then$

19) $VF2[i] = VF1[i]$

20) $elseif$

21) $VF0[i] = VF2[i]$

22) $elseif(i \neq 3\ and\ DF1[i] = DF2[i+1] = 0)then$

23) $if(VF1[i] \leqslant VF2[i+1])then$

24) $VF2[i+1] = VF1[i]$

25) $elseif$

26) $VF1[i] = VF2[i+1]$

27) $elseif(i \neq 3\ and\ DF1[i+1] = DF2[i] = 0)then$

28) $if(VF1[i+1] \leqslant VF2[i])then$

29) $VF2[i] = VF1[i+1]$

30) $elseif$

31) $VF1[i+1] = VF2[i]$

/*DENSE*/**

32) $elseif(VF1[i] = VF2[i] = 0)then$

33) $if(DF1[i] \leqslant DF2[i])then$

34) $DF2[i] = DF1[i]$

35) $elseif$

36) $DF1[i] = DF2[i]$

37) $elseif(i \neq 3\ and\ VF1[i] = VF2[i+1] = 0)then$

38) $if(DF1[i] \leqslant DF2[i+1])then$

39) $DF2[i+1] = DF1[i]$

40) $elseif$

41) $DF1[i] = DF2[i+1]$

42) $elseif(i \neq 3\ and\ VF1[i+1] = VF2[i] = 0)then$

43) $if(DF1[i+1] \leqslant DF2[i])then$

44) $DF2[i] = DF1[i+1]$

45) $elseif$

46) $DF1[i+1] = DF2[i]$

47) $endif$

48)end

– – – – –

$- - - - -$

/***CASE 3 J=0 and I=1 ***/**

/* **P0** $\rightarrow$ **P1** */

/***VOID***/**

49)$for\ i = 3\ to\ 6\ do$

50) $if(DF0[i] = DF1[i] = 0)then$

51) $VF1[i] = VF0[i]$

52) $elseif(DF0[i] = DF1[i+1] = 0\ and\ i \neq 6)then$

53) $VF1[i+1] = VF0[i]$

54) $elseif(DF0[i+1] = DF1[i] = 0\ and\ i \neq 6)then$

55) $VF1[i] = VF0[i+1]$

/***DENSE***/**

56) $elseif(VF0[i] = VF1[i] = 0)then$

57) $DF1[i] = DF0[i]$

58) $elseif(VF0[i] = VF1[i+1] = 0\ and\ i \neq 6)then$

59) $DF1[i+1] = DF0[i]$

60) $elseif(VF0[i+1] = VF1[i] = 0\ and\ i \neq 6)then$

61) $DF1[i] = DF0[i+1]$

62)end

$- - - - -$

$- - - - -$

/***CASE 4 J=2 and I=3 ***/**

/* **P2** $\rightarrow$ **P3** */

/***VOID***/**

63)$for\ i = 3\ to\ 6\ do$

64) $if(DF2[i] = DF3[i] = 0)then$

65) $if(VF2[i] \leqslant VF3[i])then$

66) $VF3[i] = VF2[i]$

67) $elseif$

68) $VF2[i] = VF3[i]$

69) $elseif(i \neq 6\ and\ DF2[i] = DF3[i+1] = 0)then$

70) $if(VF2[i] \leqslant VF3[i+1])then$

71) $VF3[i+1] = VF2[i]$

72) $elseif$

73) $VF2[i] = VF3[i+1]$

74) $elseif(i \neq 6\ and\ DF2[i+1] = DF3[i] = 0)then$

75) $if(VF2[i+1] \leqslant VF3[i])then$

76) $VF3[i] = VF2[i+1]$

77) $elseif$

78) $VF2[i+1] = VF3[i]$

/***DENSE***/**

79) $elseif(VF2[i] = VF3[i] = 0)then$
80) $if(DF2[i] \leqslant DF3[i])then$
81) $DF3[i] = DF2[i]$
82) $elseif$
83) $DF2[i] = DF3[i]$
84) $elseif(i \neq 6\ and\ VF2[i] = VF3[i+1] = 0)then$
85) $if(DF2[i] \leqslant DF3[i+1])then$
86) $DF3[i+1] = DF2[i]$
87) $elseif$
88) $DF2[i] = DF3[i+1]$
89) $elseif(i \neq 6\ and\ VF2[i+1] = VF3[i] = 0)then$
90) $if(DF2[i+1] \leqslant DF3[i])then$
91) $DF3[i] = DF2[i+1]$
92) $elseif$
93) $DF2[i+1] = DF3[i]$
94)end

– – – – –

– – – – –

/*CASE 5 J=0 and I=2 */
/* P0 → P2 */
/*VOID and DENSE*/
95) $if(DF0[3] = DF2[3] = 0)then$
96) $VF2[3] = VF0[3]$
97) $elseif(VF0[3] = VF2[3] = 0)then$
98) $DF2[3] = DF0[3]$
99) $endif$

– – – – –

– – – – –

/*CASE 6 J=1 and I=3 */
/* P1 → P3 */
/*VOID and DENSE*/
100) $if(DF1[3] = DF3[3] = 0\ and\ VF1[3] \leqslant VF3[3])then$
101) $VF3[3] = VF1[3]$
102) $elseif(DF1[3] = DF3[3] = 0\ and\ VF3[3] \leqslant VF3[3])then$
103) $DF1[3] = DF3[3]$
104) $elseif(VF1[3] = VF3[3] = 0\ and\ DF1[3] \leqslant DF3[3])then$
105) $DF3[3] = DF1[3]$
106) $elseif(VF1[3] = VF3[3] = 0\ and\ DF3[3] \leqslant DF1[3])then$
107) $DF1[3] = DF3[3]$
108) $endif$

– – – – –

– – – – –

109)end

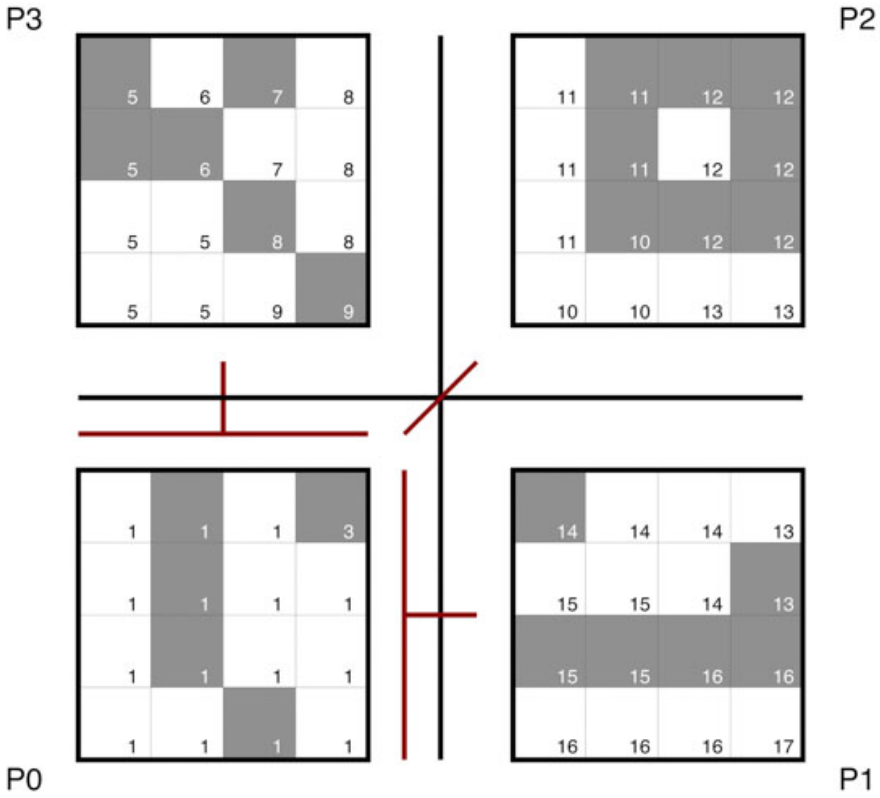

Figura 3.5: Los valores de clústering se introducen en el resto de cuadrantes partiendo de P0, CASE's 1,3 & 5 .(Ver Ejemplo teórico, sección 3.6)

3.6. Ejemplo teórico

En esta sección mostramos graficamente la evolución del proceso de clústering empleando nuestro algoritmo. Partimos de un espacio de datos ficticio, que es enrejado en 64 celdillas, las cuales son evaluadas y catalogadas en densas o vacías. Las celdas grises representan a las densas, las celdas blancas a las vacías (Figura 3.6).

A continuación, seguimos el 3-ordering dado por la curva de Hilbert con origen en la esquina inferior izquierda, (Figura 3.7) y proveemos de un etiquetado inicial a cada una de las celdas. Este etiquetado inicial se divide por cuadrantes (Figura 3.8). Siguiendo cuatro curvas de Hilbert de orden 2, una por cuadrante, vamos extendiendo el etiquetado a lo largo de los cuadrantes (Figura 3.9 y siguientes) hasta completar el procedimiento descrito anteriormente en este capítulo (Figura 3.18).

Como podemos observar en la última figura, Figura 3.18, tras practicar el proceso resultan dos componentes conexas del vacío, la numerada con la etiqueta **1** y la numerada con la etiqueta **2**. Como comentaremos en la siguiente sección, es este caso habría que aplicar otra vez el algoritmo para afinar aún más los resultados, pues, como expusimos en la introducción, buscamos que se de cumpla una condición de parada «natural», a saber, que exista solamente una etiqueta de vacío.

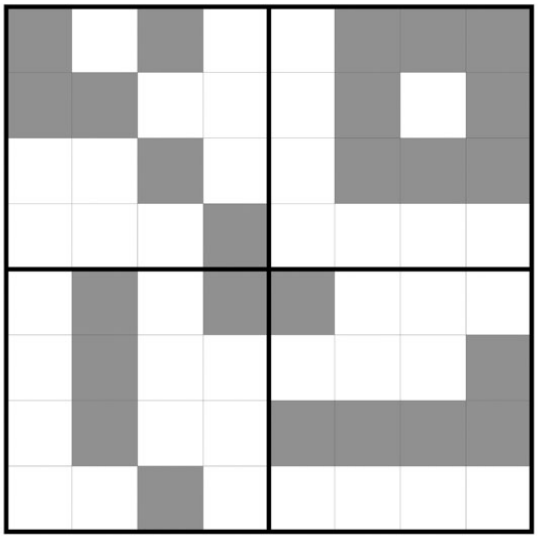

Figura 3.6: Espacio de datos dividido en 64 celdillas. Las celdas grises representan las densas y las blancas las vacías.

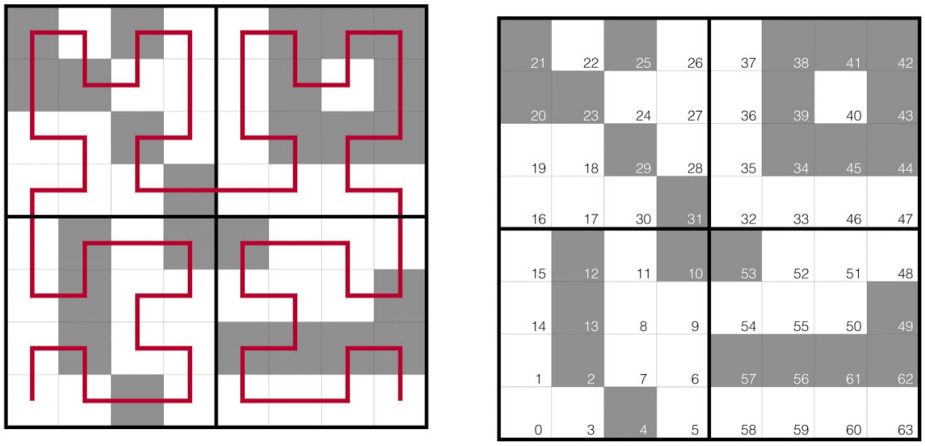

Figura 3.7: Curva de Hilbert de Orden 3 y 3-ordering asociado

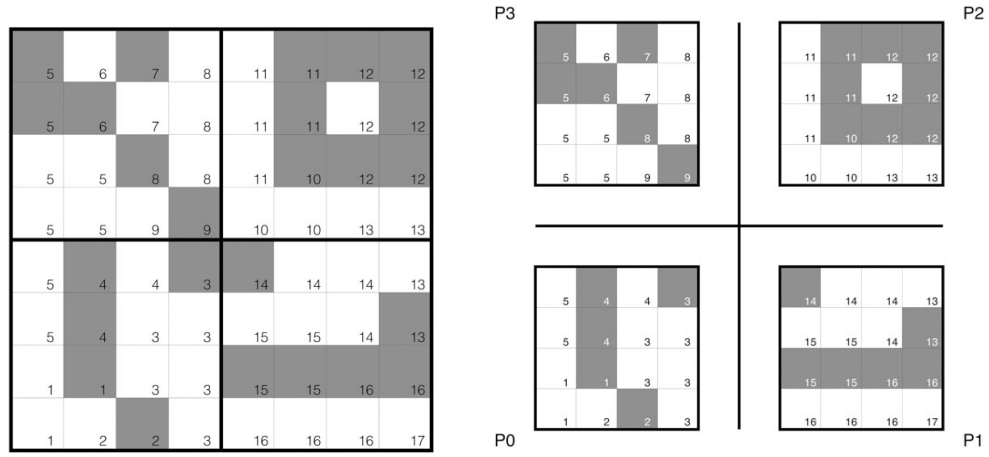

Figura 3.8: Numeración inicial dada por el 3-ordering. Espacio de datos etiquetado dividido en cuadrantes

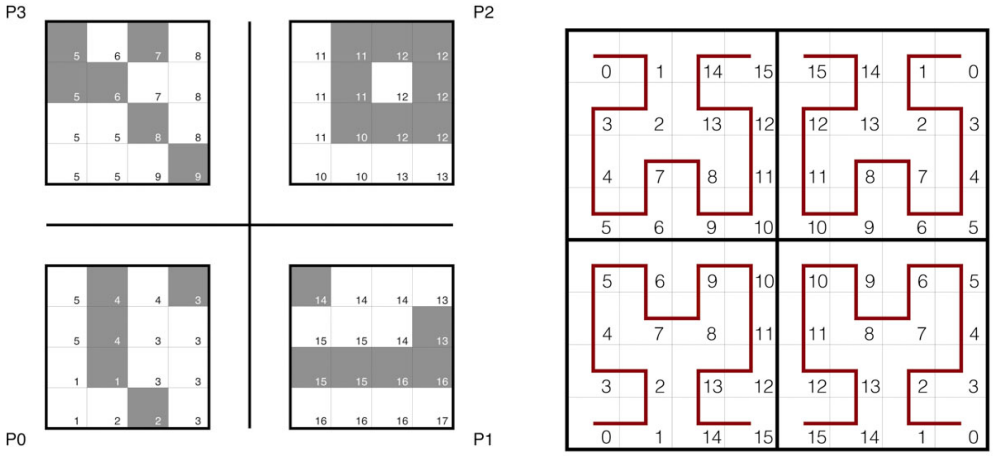

Figura 3.9: A cada cuadrante le vamos a asociar el 2-ordering correspondiente.

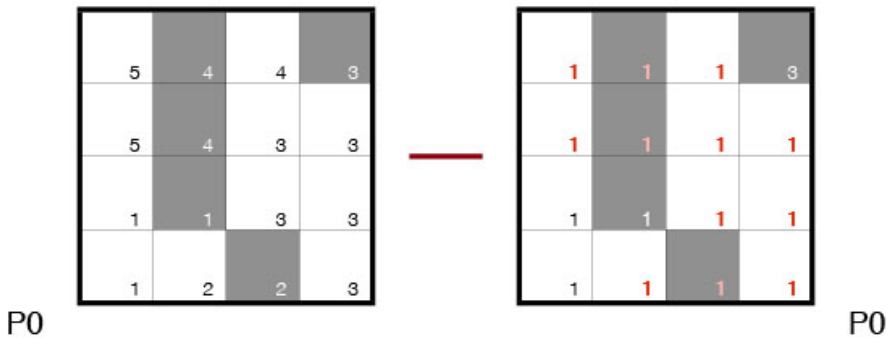

Figura 3.10: Los valores de clústering se «extienden» por P0 tras aplicar LinalDiagonal2OEvaluation y InverseLinalDiagonal2OEvaluation.

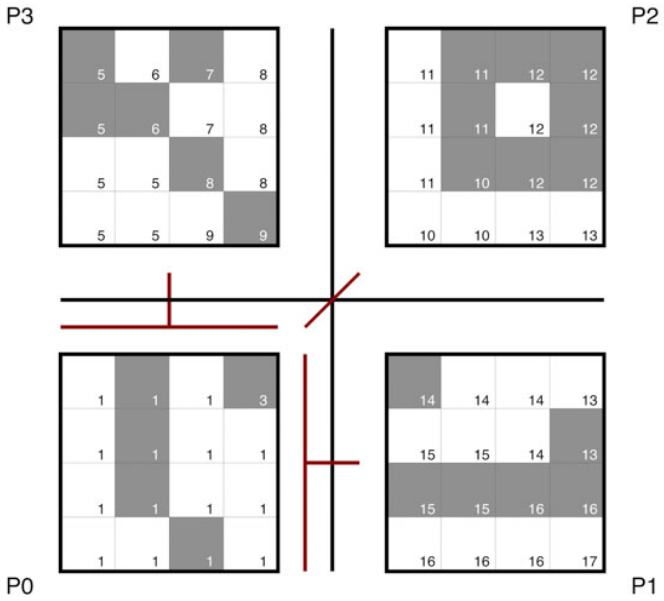

Figura 3.11: CuadrantJumps introduce los valores de clúster de P0 en el resto de cuadrantes según corresponda.

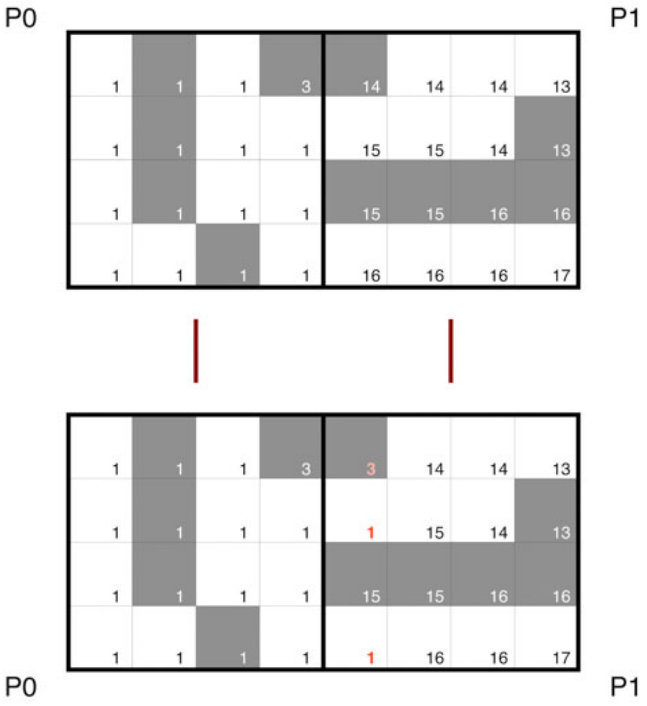

Figura 3.12: Vemos como se introducen los valores de P0 en P1.

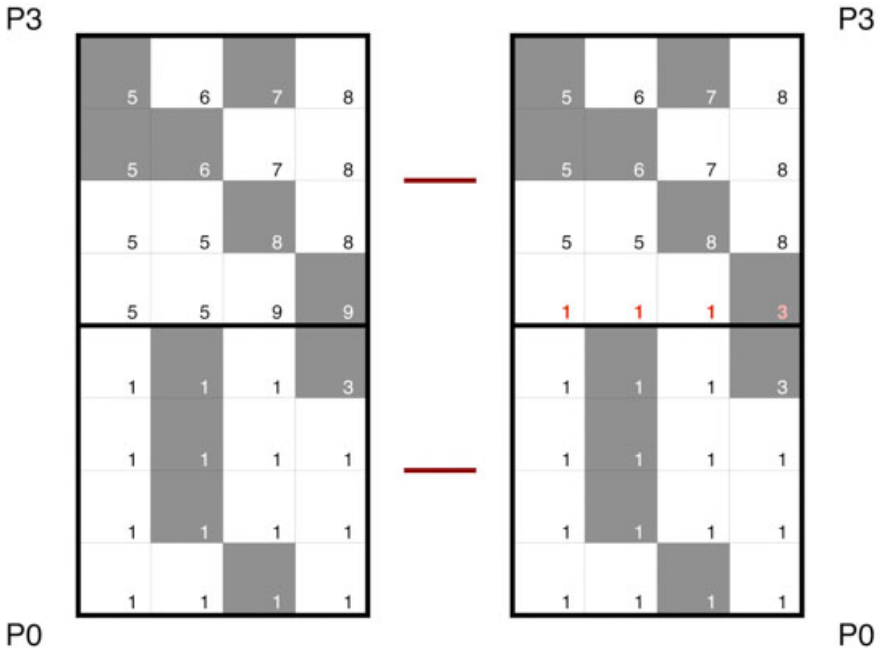

Figura 3.13: Introducción de los valores de clústering de P0 en P3.

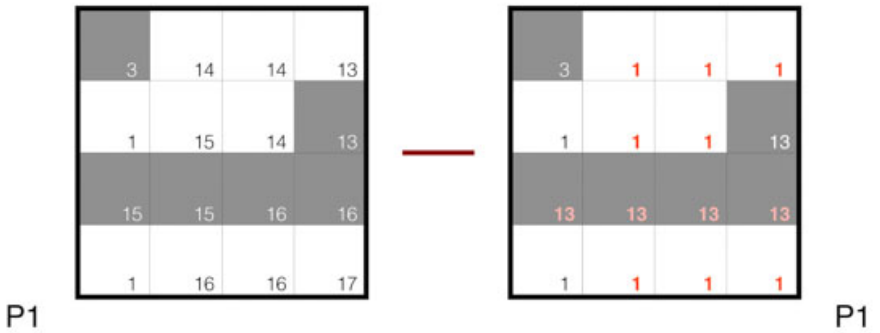

Figura 3.14: Extensión del clústering en P1.

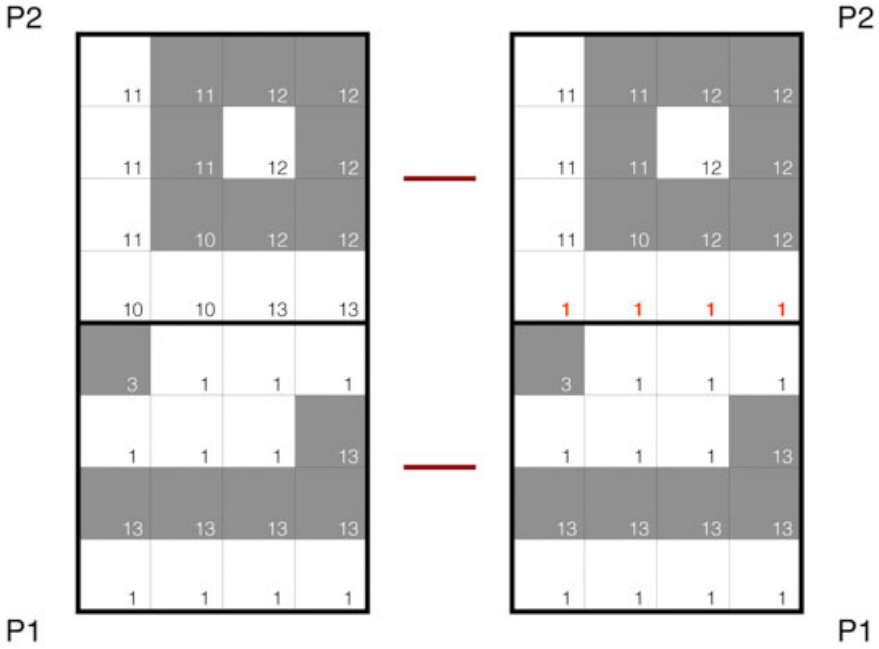

Figura 3.15: CuadrantJumps, CASE 2; J=1 & I=2

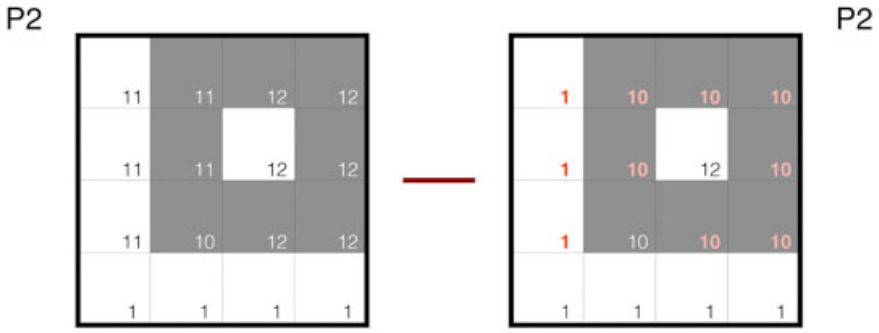

Figura 3.16: Extensión en P2

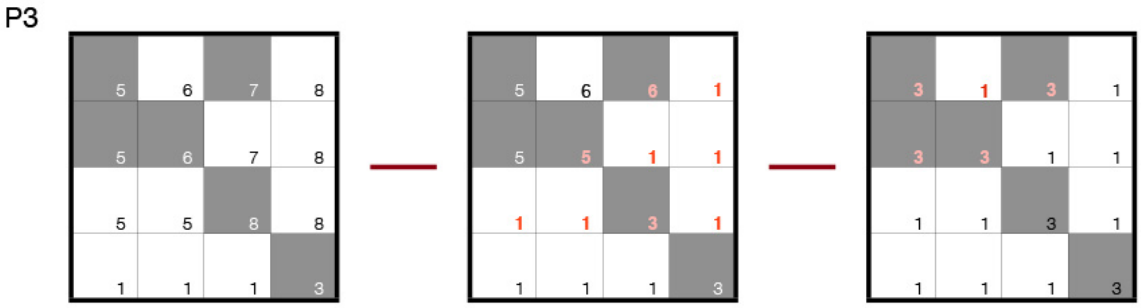

Figura 3.17: Proceso de Extensión en P3, primero tras Cuadrant Jumps CASE 1; J=0 & I=3); y posteriormente tras Cuadrant Jumps CASE 4; J=2 & I=3)

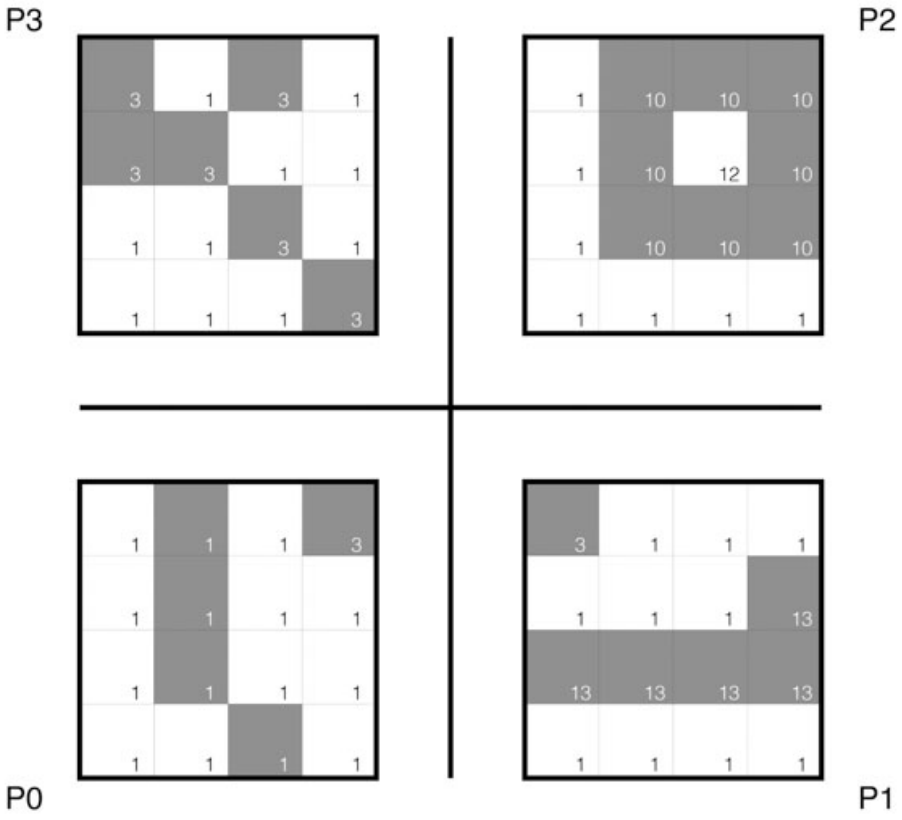

Figura 3.18: Resultado final del algoritmo

Capítulo 4

Algoritmo de ZOOM

Comentábamos en la introducción que, como proponemos un procedimiento particular con orden global 3 y órden 2 para los cuadrantes, aplicando una vez el procedimiento, el espacio de datos se ve reducido a 64 casillas, bloques o celdillas a evalúar y por tanto puede ocurrir que su resultado sea demasiado grueso. Si esto ocurre, podemos seleccionar la zona , el conjunto de celdas, que deseemos y volver a aplicarle el procedimiento.

Hasta el momento nuestro algoritmo ha accedido solamente una vez a datos, con lo que hemos minimizado una de las principales causas de retardo cuando trabajamos con grandes bases de datos. Como en el tratamiento de datos del algoritmo anterior hemos asociado a cada objeto una casilla, cuando seleccionemos las celdas donde revaluar el algoritmo, tendremos solamente que reubicar estos en las nuevas 64 celdillas. Mientras el número de objetos sea ampliamente mayor que el de celdas nos compensará usar este algoritmo.

En cuanto a la revaluación o zoom de una zona, supongamos que solamente se lo aplicamos a una celda. Hemos seleccionado una en particular que nos interesa volver a tamizar. Si le aplicamos el método una segunda vez, el tamaño de cada una de las celdas del enrejado resultante correspondería con el tamaño de casilla de una partición global de orden 6, es decir, dividir el espacio de datos en $(2^3 \times 2^3)^2 = 2^6 \times 2^6$ celdas; si lo aplicamos tres veces a una sola celda, el tamaño correspondería al de las $(2^3 \times 2^3)^3$ casillas correspondientes con el orden 9; etc. Si tomamos una zona con $m \times m$ bloques, el dominio correspondiente a estas celdillas se partirá entre 64 celdas, y cada una de estas celdillas abarcará unas hipotéticas m^2 celdas de orden 6, correspondientes con haber aplicado la curva de orden 6 a todo el espacio de datos original.

Como ya hemos dicho, el objetivo de este algoritmo es mejorar la resolución de nuestro procedimiento de clústering. Bajo ciertos supuestos puede interesarnos reevaluar un clúster en particular. En general, va a interesarnos aplicarlo a clústers densos para ver si son divisibles en subclústers. O para ver si podemos conectar una componente conexa aislada de vacío con la componente principal o más grande. También servirá para comprobar como de «vacía» es en ciertas zonas nuestra componente conexa de vacío, aunque, en general, para refinar el clústering en este sentido será mejor disminuir la $\epsilon(h)$.

A la hora de seleccionar una zona en particular, debemos plantearnos si es mejor ampliar la zona o evalúar su frontera con el exterior, entendiendo esta frontera como las celdas en contacto entre el espacio de datos que ha sido evaluado una vez y el que va a ser reevaluado una segunda vez. Ampliar la zona también exige evaluar la frontera, pero en ciertas ocasiones no supondrá cambios relevantes, por ejemplo cuando $\epsilon(h) = 0$ y la zona de ampliación sea vacía.

Tendremos que tener en cuenta que al refinar una zona, debemos guardar el número de clúster original y por otro lado guardar el propio del refinamiento para poder aunar numeraciones exteriores e interiores.

Como al reevaluar pueden aparecer nuevos clústers, miraremos si estos están en contacto con clústers ya existentes o si precisan de un nuevo número de cluster por estar aislados.

El procedimiento a seguir se compone de los siguiente pasos:

- **Paso 1**: seleccionar las celdas C(ij) que sean objeto de revisión.
- **Paso 2**: introducir sus objetos en las nuevas 64 celdas.
- **Paso 3**: hacer correr el algoritmo de la sección anterior..
- **Paso 4**: evalúar si el refinamiento ha sido suficiente. En caso de que no haya sido suficiente volver a empezar con un subgrupo de celdas de las elegidas en el paso 1
- **Paso 5**: si alguna celda de la frontera es vacía y está en contacto con una celda vacía del enrejado global, reenumerar su componente conexa. Si el vacío en el interior de las celdas reevaluadas sirve de puente entre dos componentes conexas de vacío exteriores aunar su numeración. Si en la zona reevaluada existía una componente de vacío separada, y al refinar el enrejado se ha conectado con otra componente de vacío global, ambas se reenumerarían buscando mantener la menor etiqueta.
- **Paso 6**: buscar los clúster densos en contacto con la frontera y evalúar si está en contacto con alguno denso del exterior y reenumerarlo con el número más pequeño en la numeración original. Si sirve de puente entre clústers densos exteriores reenumerar ambos con el mínimo de ambos números de clúster denso.
- **Paso 7**: volver a evalúar si el algoritmo ha alcanzado el fin perseguido.

Capítulo 5

Aplicaciones a la Astronomía

Las posibles aplicaciones a los catálogos astronómicos centrarán nuestro trabajo, como mínimo, durante el próximo año. El tiempo requerido para programar el algoritmo en un lenguaje en particular (C/C++, Python, ...), preparar una base de datos, evaluarla y filtrarla mediante el proceso de Zoom, además de la comprobación de los resultados y la busqueda de patrones, es considerable. A mayores, la naturaleza del método requiere un estudio de eficiencia del algoritmo comparado con modificaciones de los ya existentes.

El primer experimento práctico que tenemos planeado, es evaluar la base de datos OARMAC buscando posibles clústers de estrellas dobles entre sus 1800 sistemas binarios actuales. Esta evaluación nos daría información sobre la distribución de las estrellas binarias en la esfera celeste. También pensamos realizar ciertas modificaciones para desarrollar un clústering no espacial, que estudie los patrones entre las diferentes características establecidas en el catálogo, como, por ejemplo, entre las magnitudes de las estrellas del sistema y la excentricidad de la órbita; entre tipos espectrales y semiejes, periodos y exentricidades, etc.

El segundo experimento consiste en, partiendo de las zonas relativamente densas de estrellas binarias encontradas en el OARMAC, evalúar su entorno dentro de catálogos estelares mayores como el Hipparcos o el WDS. Este experimento nos puede proporcionar información sobre como es el entorno físico de los sistemas de estrellas binarias. La idea es usar la base obtenida en el OARMAC para evaluar los datos obtenidos trabajando sobre estos catálogos más amplios. También podremos comprobar, a mayor escala, los posibles patrones encontrados entre cualquier par de parámetros fisicos estudiados en el clústering no espacial.

Queremos también plantear experimentos más técnicos que nos ayuden a busquedas precisas de estrellas dobles con emisiónes tipo flare cuando las componentes se hayan próximas en su órbita. Este tipo de estudios nos permitirán elaborar teorías sobre cómo y por qué se producen estas emisiones, algo que a día de hoy sigue siendo una incógnita.

Pensamos también que este algoritmo de clústering puede ayudar a optimizar las campañas de observación astronómica que tenemos planeadas para Noviembre en el SAO (Special Astrophysical Observatory) y para la que está programada el año que viene en el BAO (Byurakan Astrophysical Observatory) en Armenia. Creemos que podemos ganar en eficiencia energética y en capacidad real de observación usando este algoritmo de clústering para detectar las zonas más interesantes de observación y para prevenir tiempos de transición entre observaciones. Los grandes telescopios, como el BTA (Big Telescope Alt-azimuth) del SAO, pueden llegar a pesar cientos de toneladas, siendo muy costoso desde el punto de vista energético su movimiento. El espejo principal del BTA tiene un diámetro de 6m. y una masa de 42 toneladas, y la estructura total del telescopio llega a las 850 toneladas repartidas a lo largo de sus 42 metros de altura;

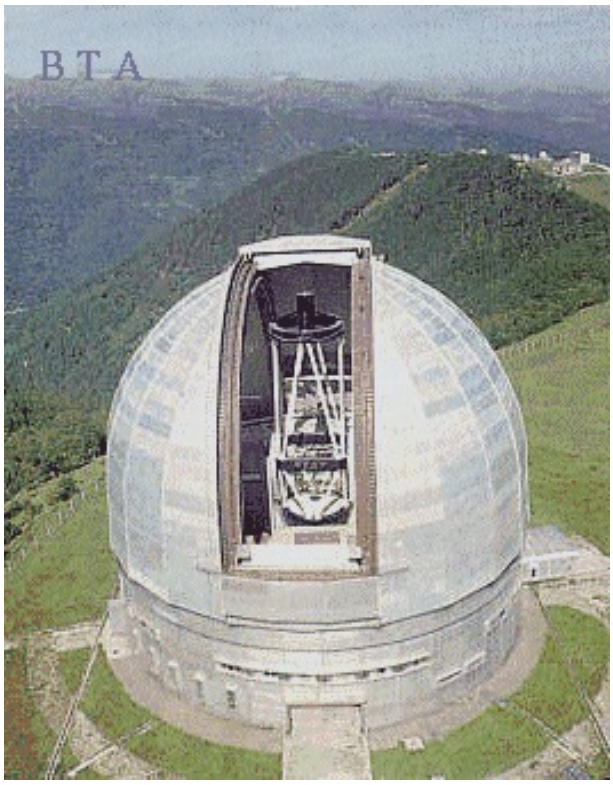

Figura 5.1: Big Telescope Alt-azimuth SAO RAS

desplazar el telescopio supone un gran gasto de energía además del gasto en tiempo. Optimizar el plan de observaciones puede llegar a suponer un gran ahorro, además de permitirnos obtener una mayor cantidad de datos en cada sesión de observación.

Por otro lado, queda por evaluar la eficiencia de nuestro método de clústering comparada con la de los otros métodos parecidos.

Capítulo 6

Futuros Trabajos

Los algoritmos desarrollados en este trabajo deben ser ahora evaluados para comprobar su eficiencia y posibles errores que se nos hayan podido pasar por alto. Su aplicación al catálogo OARMAC y al HIPPARCOS dentro de experimentos de localización y descripción de entornos de Sistemas Estelares; su posible uso en la organización y optimización de campañas de observación; su aplicación al estudio de reducción de datos provenientes de la interferometría Speckle; y otras muchas posibles aplicaciones o líneas de investigación son los retos que pensamos afrontar en el futuro.

El año que viene, como beneficiario de la beca ALRAKIS II para estudiantes de PhD, incluída dentro del programa Erasmus Mundus; espero poder continuar con estas investigaciones en el observatorio astronómico de Byurakan (BAO) en Armenia; además contamos con realizar una estancia de observación presencial en Noviembre en el Spetial Astrophysical Observatory, (SAO), de la Russian Academy of Sciences. Durante la estancia en Rusia, nuestro interés se centrará en estudiar el método de reducción de datos que se aplica a la interferometría Speckle y ver si es mejorable tanto con los nuevos métodos de Data Mining, como los tratados en este trabajo, como con los que puedan aparecer en los meses venideros.

Apéndice A

Catálogos Astronómicos

A.1. OARMAC

La primera versión del OARMAC fue publicado en hace ahora 12 años [33] coincidiendo con la primera versión on-line. Esta primera versión contenía inicialmente 1.545 órbitas de 1.208 sistemas. Los autores de este catálogo son: J.A. Docobo, J.F. Ling, J.M. Costa, M.T. Costado y P. Magdalena, con prefacio de P. Couteau.

El catálogo se suele actualizar una vez al año. En sus primeros tiempos por J.A. Docobo, J.F. Ling, C. Prieto, M.T. Costado y P. Magdalena. Después por J. A. Docobo, J. F. Ling, C. Prieto and M. Brea. Y actualmente, el equipo de astrónomos que ha estado al cargo lo componen J. A. Docobo, J. F. Ling y P. P. Campo [34].

La última versión se actualizó el 4 de Octubre del 2012, contiene 2188 órbitas de 1776 sistemas, estando previsto hacer una nueva a finales de Julio de 2013.

A.2. Washington Double Star Catalog

El catálogo Washington Double Star (WDS) [35], mantenido por el Observatorio Naval de los Estados Unidos, es la más importante base de datos en información astrométrica de estrellas dobles y múltiples. El catálogo WDS alberga, de las componentes de 126211 sistemas, posiciones (J2000), códigos, épocas, ángulos de posición, separaciones angulares, magnitudes, tipos espectrales, movimientos propios y, cuando es posible, los números Durchmusterung y anotaciones obtenidas en sus descubrimientos.

A.3. Hipparcos Space Astrometry Mission

ESA's Hipparcos space astrometry mission fue un proyecto europeo pionero el cual reporó las posiciones de más de cien mil estrellas con gran precisión, y más de un millón de estrellas con menor precisión.

Puesto en órbita en Agosto de 1989, el satélite Hipparcos observó con éxito la esfera celeste durante tres años y medio antes de que las operaciones concluyesen en Marzo de 1993. Las observaciones realizadas por la instrumentación principal generaron el catálogo HIPPARCOS [36], 118.218 estrellas catalogadas

con gran precisión. Un mapa auxiliar determina la posición de muchas otras estrellas que, aun teniendo un grado menor de precisión que las anteriores, desde entonces conocemos con una precisión desconocida hasta el momento. En el catálogo Tycho [36] se encuentran los datos de 1.058.332 estrellas. El catálogo Tycho 2, completado en el año 2000, nos aporta información de un total de 2.539.913 estrelas, incluyendo el 99 % de todas las estrellas de magnitud inferior a 11, hasta 100.000 veces más ténues que Sirius.

Los resultados finales de la misión Hipparcos -los catálogos HIPPARCOS y TYCHO- fueron publicados en Junio de 1997. Estos fueron elaborados bajo la responsabilidad de una gran cantidad de equipos de científicos colaboradores de la ESA. Los líderes del consorcio fueron el Dr Lennart Lindegren (Lund, Sweden: NDAC) y el profesor Jean Kovalevsky (Grasse, France: FAST) responsables conjuntos del catálogo HIPPARCOS; el professor Erik H¿g (Copenhagen, Denmark: TDAC) a cargo del TYCHO Catalogue; y la Dr Catherine Turon (Meudon, France: INCA) responsable del HIPPARCOS Input Catalogue.

A.4. Gaia

La European Space Agency (ESA) lanzará, a finales de este año, el satélite Gaia con el propósito de formar un mapa 3D que contenga parte de nuestra galaxia, tanto estrellas como ciertos planetas y asteroides. Su objetivo inicial consiste en censar mil millones de estrellas, esto supone analizar un 1 % de la población de la Vía Láctea. Interesa estudiar concienzudamente su posición y paralaje así como evaluar sus propiedades físicas, como la naturaleza binaria o múltiple, su temperatura, luminosidad o composición química. Cada una de estas estrellas será observada una media de 70 veces durante los 5 años que dura la misión, se supone que realizará del orden de 40 millones de observaciones por día, lo cual nos da una idea del volumen de datos que generará esta importante misión espacial.

Gaia se basa en el legado científico y tecnológico de la misión espacial Hipparcos y recoge el testigo en la investigación sobre de que está hecho y como se formó el pequeño trozo de Universo que nos rodea.

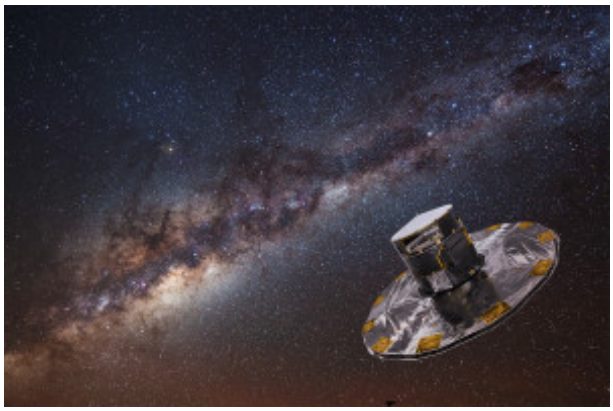

Figura A.1: Gaia censando las estrellas de la Vía Láctea desde el Punto L2 de Lagrange, a 1,5 millones de Km. de la Tierra en dirección opuesta al Sol.

Conclusiones

En el presente TFM, se ha buscado mostrar las posibilidades que el clústering abre dentro del estudio de las grandes bases de datos astronómicos. En consonancia, hemos desarrollado un algoritmo propio que evalúa y etiqueta tanto las celdas que consideramos densas, es decir, aquellas que contienen, como mínimo, un número prefijado de objetos; como aquellas celdas, que por no llegar a esta cantidad, consideramos vacías.

Esto ha supuesto aplicar una técnica de utilidad testada y corroborada en campos tan punteros como la tecnología *Wireless* o la investigación en genética. Dados los artículos y estudios a los que hemos tenido acceso, la utilización de la HSFC, como base del indexado, nos pareció la opción más adecuada. Las modificaciones realizadas con respecto a métodos anteriores surgen de las restricciones propias del campo de aplicación, la Astronomía.

La aplicación a un ejemplo teórico sirve para comprobar que la problemática propia de las curvas de Hilbert, en cuanto a los saltos de numeración entre celdas contiguas, ha sido solventada. La extensión del etiquetado se muestra como una estrategia satisfactoria a la hora de buscar el clústering tanto lineal como diagonalmente. De la misma manera, el proceso de evaluación de resultados que hemos denominado ZOOM, por razones obvias, parece complementar satisfactoriamente el algoritmo y nos permite un refinamiento exhaustivo de los resultados.

Bibliografía

[1] «An Efficient Curve-based Clustering Strategy for Large Spatial Databases», by Yun-Tai Lu. *A thesis submited to the Faculty of National Sun Yat-sen University.*

[2] «Efficient Range Query Using Multiple Hilbert Curves», by Ying Jin (Cold Spring Harbor Lab), Jing Dai (IBM T. J. Watson Research Center) and Chang-Tien Lu (Virginia Polytechnic Institute and State University), USA.

[3] «Large Efficient Searching Algorithms for Multi-dimensional Space Data Using Hilbert Space-filling Curves», by Chih-Sheng Chen, Jan-Yie Liang, Yi-Kun Lee, Min-Hsuan Fan, Chua-Huang Huang. *Journal of Information & Computational Science Vol. 8, No. 1, pp. 41Ð50, 2011.*

[4] «Über die stetige Abbildung einer Linie auf ein Flächenstück», by D. Hilbert. *Math. Ann. 38 (1891), 459Ð460.*

[5] «On Simulations of Galaxy Dynamics and their Application to Physics Education», by Jakub Schwarzmeier. *A dissertation submitted to the University of West Bohemia in partial fulfillment of the requirement for the degree of doctor in Theory of Education in Physics.* Supervisor RNDr. Miroslav Randa, Ph.D. Date of submission: April 30, 2007.

[6] «Spatial Queries inWireless Broadcast Systems», by Baihua Zheng (Singapore Management University), Wang-Chien Lee (The Penn State University) and Dik Lun Lee (Hong Kong University of Science and Technology).

[7] «Using Space-filling Curves for Multi-dimensional Indexing», by J. K. Lawder and P. J. H. King (School of Computer Science and Information Systems, Birkbeck College, University of London).

[8] «DSI: A Fully Distributed Spatial Index for Location-based Wireless Broadcast Services», by Wang-Chien Lee (The Pennsylvania State University).

[9] «A Hilbert Curve-Based Distributed Index for Window Queries inWireless Data Broadcast Systems», by Jun-Hong Shen and Ye-In Chang (Department of Computer Science and Engineering, National Sun Yat-Sen University).

[10] «Detección de Pares de Movimiento Propio Común Mediante Minería de Datos», by Blanca Collado Iglesias, Antonio Javier Fernández Sánchez, and Sara Pozuelo González; directed by Rafael Caballero Roldán. *Proyecto de Sistemas Informáticos, curso 2009/2010. Universidad Complutense Madrid.*

[11] «Data Mining and Machine Learning in Astronomy», by Nicholas M. Ball (Herzberg Institute of Astrophysics, National Research Council, Canada) and Robert J. Brunner (Department of Astronomy, University of Illinois at Urbana-Champaign, USA).

[12] «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise», by M. Ester, H. P. Kriegel, J. Sander, and X. Xu. *Proc. of the 2nd Int. Conf. on KDD, pp.*

226-231, 1996.

[13] «Finding Groups in Data: An Introduction to Cluster Analysis», by L. Kaufman and P. J. Rousseuw. *John Wiley & Sons, Inc., New York, 1990.*

[14] «Efficient and Effective Clustering Methods for Spatial Data Mining», by R. T. Ng and J. Han. *Proc. of the 20th VLDB Conf., pp. 144-155, 1994.*

[15] «Implementing Agglomerative Hierarchical Clustering Algorithms for Use in Document Retrieval», by E. M. Voorhees. *Information Proc. & Management, pp. 465-476, 1986.*

[16] «BIRCH: A Efficient Data Clustering Method for Very Large Databases», by T. Zhang, R. Ramakrishnan, and M. Livny. *ACM SIGMOD Int. Conf. on Management of Data, pp. 103-114, 1996.*

[17] «ROCK: A Robust Clustering Algorithm for Categorical», by S. Guha, R. Rastogi, and K. Shim. *Int. Conf. on Data Eng., pp. 512-521, 1999.*

[18] «CURE: An Efficient Clustering Algorithm for Large Databases», by S. Guha, R. Rastogi, and K. Shim. $InformationSystems, Vol$,26, No,1, pp,35 − 58, $March$2001.

[19] «CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling», by G. Karypis, E. H. Han, and V. Kumar. $IEEETrans.onComputer, Vol$,32, No,8, pp,68 − 75, Aug,1999.

[20] «Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications», by R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. $ACMSIGMODConf., pp$,94 − 105, 1998.

[21] «OPTICS: Ordering Points To Identify the Clustering Structure», by M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. *ACM SIGMOD Conf., pp. 49-60, 1999.*

[22] «Clustering Gene Expression Patterns», by A. Ben.Dor, R. Shamir, and Z. Yakhini. *Proc. of the 3rd Annual Int. Conf. on Computational Molecular Biology, pp. 33-42, 1999.*

[23] «WaveCluster: A MultiResolution Clustering Approach for Very Large Spatial Databases», by G. Sheikholeslami, S. Chatterjee, and A. Zhang. *Proc. of the 24th VLDB Conf., pp. 428-439, 1998.*

[24] «STING: A Statistical Information Grid Approach to Spatial Data», by W. Wang, J Yang, and R. Muntz. *Proc. of the 23th VLDB Conf., pp. 186-195, 1997.*

[25] «Sur une courbe, qui remplit toute une aire plane», by G.Peano. *Mathematische Annalen 36 (1890), 157Ð160.*

[26] «A Class of Data Structures for Associative Searching», by Jack A. Orestein and T. H. Merret. *Proc. Symp. on PODS., pp. 181-190, 1984.*

[27] «Spatial Query Processing in an Object-Oriented Database System», by Jack A. Orestein. *Proc. ACM SIGMOD Int. Conf. on Management of Data. pp. 326-336. 1986.*

[28] «Pulse code communication», by F. Gray. 17 de marzo de 1953 (archivado en nov. 1947). *Patente USPTO n.º 2632058.*

[29] «Gray Codes for Partial Match and Range Queries», by Christos Faloutsos, *IEEE Trans on Software Eng., Vol. 14, No. 10, pp. 1381-1393, Aug. 1988.*

[30] «Fractals for Secondary Key Retrieval», by Christos Faloutsos and Shari Roseman. *ACM SIGACT-SIGMOD-SIGART Symposium on PODS, pp. 247-252, 1989.*

[31] «Analysis of the Clustering Properties of the Hilbert Space-Filling Curve», by BongKi Moon, H.V. Jagadish, Christos Faloutsos, and Joel H. Saltz. *IEEE Trans. on Knowledge and Data Eng., Vol. 13, No. 1, pp. 124-141, Jan. 2001.*

[32] «Vertex-Labeling Algorithms for the Hilbert Space-Filling Curve», by John J. Bartholdi III,and Paul Goldsman. January 11, 2000.

[33] By J. A. Docobo, J. F. Ling, C. Prieto, J. M. Costa, M. T. Collado, and P. Magdalena. *Acta Astronómica, Vol. 51, 353-356, 2001.*

[34] By J. A. Docobo, et al. 2013 version, Catalogue of Orbits and Ephemerides of Visual Double Stars,
http://www.usc.es/astro/catalog.htm

[35] By Mason, B. D., Wycoff, G. L., & Hartkopf, W. I. 2009, The Washington Double Star Catalog,
http://ad.usno.navy.mil/wds/wdstext.html#intro

[36] ESA 1997, The Hipparcos and Tycho Catalogues (Noordwijk: ESA Pub. Div. ESTEC).
http://www.rssd.esa.int/index.php?project=HIPPARCOS

[37] http://dblp.uni-trier.de/db/

[38] http://www.wikipedia.org/

[39] Figure 1.1: Ejemplo de dendrograma
http://www.nonlinear.com/support/progenesis/samespots/faq/dendrogram.aspx

[40] Figura 2.2: Curvas de Peano de orden 1, 2, 3 y 4
http://stackoverflow.com/questions/17152092/cuda-space-filling-curves-and-dividing-work-between-processors

[41] Figure 2.3: Curvas de Hilbert hasta orden 6
http://en.wikipedia.org/wiki/Space-filling_curve

[42] Figure 3.1: En esta figura podemos observar las numeraciones asociadas a los tres primeros órdenes de la curva de Hilbert. Los números dentro de cada celda representan el h-ordering.
http://www.thirstyturtle.be/

[43] Figure 5.1: Big Telescope Alt-azimuth SAO RAS
http://www.sao.ru

[44] Figure A.1: Gaia censando las estrellas de la Vía Láctea desde el Punto L2 de Lagrange, a 1,5 millones de Km. de la Tierra en dirección opuesta al Sol. Credit: ESA/ATG medialab; background image: ESO/S. Brunier
http://sci.esa.int/gaia

[45] http://www.google.es

El resto de las figuras fueron realizadas por Andrés Fraga siguiendo los bocetos del autor. Andrés es profesor asociado a la asignatura de Fotografía Contemporánea en la Facultad de Ciencias de la Comunicación de la USC.