UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
MAESTRÍA EN CIENCIAS (FÍSICA)
FÍSICA ESTADÍSTICA Y SISTEMAS COMPLEJOS

MÉTODO DUAL GENERALIZADO DESCENTRALIZADO: ALGORITMO EFICIENTE PARA ESTUDIAR LA DINÁMICA EN AMBIENTES CUASIPERIÓDICOS.

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS (FÍSICA)

PRESENTA:
ALAN RODRIGO MENDOZA SOSA

DIRECTOR DE TESIS
DR. RICARDO ATAHUALPA SOLÓRZANO KRAEMER
FACULTAD DE CIENCIAS, UNAM

COMITÉ TUTOR
DR. DAVID PHILIP SANDERS
FACULTAD DE CIENCIAS, UNAM

DR. CHUMIN WANG CHEN
INSTITUTO DE INVESTIGACIONES EN MATERIALES, UNAM

CIUDAD UNIVERSITARIA, CIUDAD DE MÉXICO, MARZO DE 2021

"Life before Death.
Strength before Weakness.
Journey before Destination."
Brandon Sanderson

Reconocimientos

A mis padres Salvador Mendoza Gómez y Rosa Martha Sosa Guzmán por todo el apoyo y amor brindado, por la paciencia y confianza en cada uno de mis proyectos y por acompañarme a lo largo de estos 25 años de vida, permitiéndome extender mis alas para volar alto, pero siempre atentos por si las fuerzas me fallan.

A mi hermana Martha Anahí Mendoza Sosa por ser el motor de crecimiento durante mi infancia y la motivación constante para superarme cada día.

A Ricardo González Almanza, mi mejor amigo, por más de 7 años compartiendo gustos, proyectos, debatiendo ideas y jugando hasta altas horas de la madrugada juegos en línea.

Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea el resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Alan Rodrigo Mendoza Sosa. Ciudad de México, 2020

Resumen

La presente tesis representa una investigación orientada al desarrollo de un algoritmo computacional eficiente, basado en el método dual generalizado, que nos permita estudiar una amplia variedad de arreglos cuasiperiódicos con simetría rotacional $N \in \mathbb{Z}^+$ alrededor de puntos alejados del origen. A partir de la implementación de dicho algoritmo se presentan los resultados correspondientes al análisis de la difusión en un sistema cuasiperiódico de simetría pentagonal con un potencial de interacción de esferas duras así como un análisis de la función de distribución radial en sistemas cuasiperiódicos de alta simetría. Con el objetivo de brindar un contexto sólido al trabajo presentado se expone en el primer capítulo una recopilación bibliográfica e histórica referente a los sistemas cuasiperiódicos, desde un punto de vista tanto matemático como físico.

This thesis represents an investigation aimed at the development of an efficient computational algorithm, based on the generalized dual method, which allows us to study a wide variety of quasiperiodic arrangements with rotational symmetry $N \in \mathbb{Z}^+$ around points far from the origin. From the implementation of this algorithm, the results corresponding to the analysis of diffusion in a quasiperiodic pentagonal symmetry system with a hard sphere potential are presented, as well as an analysis of the radial distribution function in quasiperiodic systems of high symmetry. In order to provide a solid context to the work presented, the first chapter presents a bibliographic and historical compilation regarding quasiperiodic systems, from both a mathematical and physical point of view.

Índice general

Capítulo 1

Introducción

La ciencia, en particular la física, ha sido encomendada a la tarea de explicar con la mayor precisión posible los fenómenos presentes en el mundo que nos rodea, extendiendo su dominio de aplicación desde los componentes más básicos de la materia (aquellas partículas tan pequeñas que nuestros ojos, incluso con el apoyo de diversos instrumentos de amplificación, no pueden ver) hasta el universo mismo, estudiando su expansión, su origen y los cuerpos en él contenidos tales como los planetas, las estrellas y las galaxias, por mencionar algunos.

Dicha labor está motivada por la creencia (respaldada fuertemente por un sinfín de experimentos realizados a lo largo de la historia de la humanidad) de que existe un orden subyacente a la realidad que percibimos, de que hay un por qué y un cómo a todo aquello que observamos y, más importante aún, de que los resultados obtenidos son reproducibles sin importar el tiempo ni el lugar en el que se desarrollen los experimentos correspondientes. Sin embargo, como todo proceso creativo, la búsqueda de las teorías y leyes que dan respuesta a las preguntas planteadas está sujeta a las ideas predominantes de la época respecto a lo estético y sobre todo a lo simple (buscamos teorías que expliquen lo "más" con lo "menos") y por lo mismo resulta común que, cada vez que surge evidencia experimental que rompe con estas ideas, aparezca una fuerte resistencia a abandonar o expandir las teorías vigentes hasta ese momento.

Tal es el caso de los cuasicristales descubiertos en 1984 por D. Shechtman (ganador del Premio Nobel de Química en el 2011 por dicho descubrimiento), I. Blech, D. Gratias y J. W. Cahn [1], quienes reportaron la presencia de un grupo de simetría de punto icosaédrico (prohibido por la cristalografía clásica basada en la hipótesis de que todo cristal debe tener una distribución periódica de sus átomos) en el patrón de difracción de una aleación de aluminio y manganeso. Este resultado repercutía directamente en la manera en la que se entendían a los sólidos cristalinos y fue duramente criticado, sin embargo la cantidad de resultados experimentales reportados por otros investigadores sobre esa misma línea de investigación aportaba más y más evidencia a favor de la existencia de estas estructuras atómicas (las cuales pasaron a denominarse estructuras

cuasiperiódicas) presentes no sólo en aleaciones metálicas, sino en diversos sistemas tales como calcogenoides, coloides, entre otros [2, 3, 4]. En el capítulo 2 se ahondará en los detalles, tanto matemáticos como experimentales, sobre esta clase de sistemas.

Una vez aceptada la existencia de las estructuras cuasiperiódicas en diversos sistemas físicos, era necesario contar con un modelo matemático para las mismas que permitiera a los físicos teóricos estudiar las propiedades presentes en dichos sistemas; sobre esta cuestión surgieron distintos enfoques para abordar el problema, siendo el enfoque computacional (esto es, la creación de algoritmos computacionales capaces de generar arreglos cuasiperiódicos para su posterior estudio a partir de experimentos simulados por computadora) el que se abordará a lo largo de los capítulos 3 y 5, en los que presentaremos las cuestiones teóricas referentes a un algoritmo eficiente basado en el Método Dual Generalizado así como su implementación en un código escrito en el lenguaje de programación Julia.

En la actualidad, la capacidad de cómputo de las computadoras personales, así como su capacidad de almacenamiento, son lo suficientemente altas y a un costo relativamente bajo como para poder desempeñar labores de investigación fuera de los centros especializados presentes en los institutos, universidades y empresas privadas; sin embargo, por muy altas que sean las capacidades técnicas de las computadoras a las que se tenga acceso, estas tienen un límite y un algoritmo que tarde años en ejecutarse no tiene utilidad práctica, es por ello que resulta vital el generar algoritmos eficientes que nos permitan obtener resultados numéricos en cuestión de días o, en su defecto, semanas.

Sobre este punto, en el capítulo 4, se presentan las ideas, la teoría y los métodos de construcción del teselado de Voronoi, el cuál es un algoritmo que nos permite separar el plano por regiones, de tal forma que dado un conjunto de sitios, la región que contiene a cada uno de ellos es tal que cualquier punto dentro de ella dista menos del sitio contenido en esa región que de cualquier otro sitio; en otras palabras, nos permite separar el plano de tal forma que, localizados en cualquier punto, podemos identificar qué sitio nos queda más cerca. Este algoritmo juega un papel fundamental en el desarrollo del nuestro para reducir la cantidad de memoria requerida al generar los arreglos cuasiperiódicos, así como identificar fácilmente qué polígonos de un arreglo cuasiperiódico colindan con algún polígono dado, lo que permite estudiar la dinámica de una partícula dentro de un sistema cuasiperiódico de manera eficiente.

En el capítulo 6 presentamos algunos resultados numéricos producto de implementar el código desarrollado durante el capítulo 5 (cuyo pseudocódigo es presentado en un apéndice) en diversos contextos tales como el cálculo de la difusión de un sistema cuasiperiódico tipo Penrose con un potencial de interacción de esferas duras o el cálculo de la función de distribución radial para diferentes arreglos cuasiperiódicos. Por último, en el capítulo 7 presentamos las conclusiones de nuestro trabajo.

En resumen, la presente tesis es un esfuerzo para generar herramientas computacionales orientadas al estudio de ambientes cuasiperiódicos, en particular se presenta un algoritmo basado en el Método Dual Generalizado para producir vecindades alrededor de un punto arbitrario en el plano de arreglos cuasiperiódicos, así como estudiar la dinámica en dichos sistemas dado el potencial de interacción entre los sitios del arreglo cuasiperiódico y una partícula que se desplaza en él.

Capítulo 2

Ambientes Cuasiperiódicos

El concepto de periodicidad, como algo que se repite de manera infinita a lo largo del espacio o el tiempo, es uno de los conceptos matemáticos que, una vez definido, nos resulta hasta cierto punto fácil de imaginar; pensemos por ejemplo en el ciclo del día y la noche, que si bien estrictamente hablando no es un fenómeno periódico (la duración de cada uno de estos dos eventos varía al cambiar las estaciones del año), se presenta ante nosotros con la suficiente regularidad para poder interpretarlo como un ciclo periódico en el tiempo. Por otro lado, para el caso espacial, imaginemos la distribución de las celdas hexagonales presentes en un panal de abejas, en donde a pesar de no poseer una extensión infinita, es fácil identificar el patrón que se repite (salvo algunas imperfecciones) a lo largo de toda su superficie.

De manera formal, y para el caso unidimensional, definimos a las funciones periódicas de la siguiente manera:

- **Función periódica:** Sea f una función definida en $X \subset \mathbb{R}$, decimos que la función f es periódica con un periodo $T \neq 0$ si para cada $x \in X \subset \mathbb{R}$ los números $x + T$ y $x - T$ también pertenecen a X, de modo que se cumple la igualdad $f(x \pm T) = f(x)$. [5]

A partir de la definición previa podemos notar que las funciones periódicas en una dimensión forman un subconjunto muy limitado de todas las funciones de $\mathbb{R}$ en $\mathbb{R}$, subconjunto que cabe mencionar, no es cerrado bajo la suma usual de funciones. La condición fundamental que define a este subconjunto (esto es que exista un número real distinto de cero tal que $|f(x \pm T) - f(x)| = 0 \ \forall x \in X$) es precisamente lo que nos impide modelar con total rigor a los dos ejemplos previos (el ciclo del día y la noche y la distribución hexagonal presente en un panal de abejas) como fenómenos u objetos periódicos.

Centrémonos de momento en el caso del ciclo del día y la noche, dado que existen estas ligeras variaciones en la duración de los días conforme la Tierra va orbitando alrededor del sol, no nos es posible encontrar un periodo T para el cual se satisfaga la igualdad mencionada. Sin embargo, si estas variaciones son pequeñas, podemos aspirar a

que existan algunos valores $\tau \in \mathbb{R}$ para los que la igualdad presente en la definición de las funciones periódicas se satisfaga aproximadamente con un grado arbitrario de precisión. Esta idea es la que da lugar a las denominadas **funciones casiperiódicas** (almost periodic functions), las cuáles fueron estudiadas por el matemático danés Harald Bohr quien creó y desarrolló la teoría matemática para dichas funciones entre los años 1923 y 1925 [6].

2.1. Definición de casiperiódico y cuasiperiódico

Pensemos en una función $F(x)$ que no es periódica, pero cuya distancia entre $F(x)$ y $F(x+\tau)$, para algún valor $\tau \in \mathbb{R}$, es muy pequeña, es decir, los valores $F(x)$ y $F(x+\tau)$ son **casi** iguales. De una manera un tanto informal decimos que una función $F(x)$ es **casiperiódica** (almost periodic function) si cumple que para toda $\epsilon > 0$ existe $\tau \in \mathbb{R}$ tal que:

$$|F(x+\tau) - F(x)| \leq \epsilon, \quad \forall x \in \mathbb{R}, \tag{2.1}$$

esta condición contiene la esencia de las funciones casiperiódicas salvo el hecho de que para valores de ϵ relativamente grandes (no tan cercanos a cero) el valor de τ correspondiente podría crecer exponencialmente, lo cuál no queremos que ocurra. Para evitarlo pedimos que el conjunto de los τ sea relativamente denso en $\mathbb{R}$.

Las nociones anteriores se formalizan de la siguiente manera:

- **Casiperíodo:** Sea $f : \mathbb{R} \to \mathbb{C}$ una función continua en $\mathbb{R}$. $\tau = \tau_f(\epsilon) \in \mathbb{R}$ es llamado un número de traslación o un casiperíodo de f correspondiente a $\epsilon > 0$ si $|f(x+\tau) - f(x)| \leq \epsilon, \quad \forall x \in \mathbb{R}$.
- **Conjunto relativamente denso en $\mathbb{R}$:** Un subconjunto $A \subset \mathbb{R}$ es relativamente denso en $\mathbb{R}$, si existe $l > 0$ tal que cualquier intervalo de longitud l contiene al menos un número de A.

De esta forma, definimos a las funciones casiperiódicas como:

- **Función casiperiódica:** Una función continua $f : \mathbb{R} \to \mathbb{C}$ es llamada casiperiódica, si dado $\epsilon > 0$ el conjunto de todos los números de traslación de f correspondientes a ϵ ($\{\tau_f(\epsilon)\}$) es relativamente denso en $\mathbb{R}$. En otros términos, una función continua f es casiperiódica si dado $\epsilon > 0$, existe $l = l(\epsilon) > 0$ tal que cualquier intervalo de longitud l contiene al menos un número de traslación τ de f correspondiente a ϵ. [6]

2.1.1. Funciones Cuasiperiódicas

Un resultado conocido en el análisis de Fourier es que toda función continua puede ser expresada como una serie infinita de funciones armónicas con periodos distintos. Cuando la función de interés se obtiene a partir de una suma finita de dichas funciones armónicas el resultado puede ser periódico o casiperiódico. Generalizando esta idea de funciones casiperiódicas obtenidas a través de sumas finitas de funciones periódicas (no necesariamente armónicas) para las cuales los periodos de todas ellas son inconmensurables entre sí, construimos una nueva categoría de funciones, a las cuales denominamos **funciones cuasiperiódicas** (quasiperiodic functions). Formalmente la definición de una función cuasiperiódica es:

- **Función cuasiperiódica:** Una función $f(t)$ es cuasiperiódica si cumple que $f(t) = F(t, ..., t)$ para alguna función continua $F(t_1, t_2, ..., t_n)$ de n variables que es periódica con respecto a $t_1, t_2, ..., tn$ con períodos $w_1, w_2, ..., w_n$ respectivamente. Todos los períodos $w_1, w_2, ..., w_n$ deben ser estrictamente positivos e inconmensurables entre sí.

Una definición equivalente se puede encontrar en [7]. Cabe resaltar que actualmente no existe consenso sobre la definición de funciones cuasiperiódicas, sin embargo para los fines del presente trabajo emplearemos la definición anterior.

2.2. Teselados aperiódicos y cuasicristales

Tras haber definido en la sección anterior lo que entendemos por una función cuasiperiódica (la cual es, a grandes rasgos, una proyección desde un espacio n-dimensional a un espacio unidimensional de una función periódica en cada una de las n dimensiones) se podría pensar que los objetos cuasiperiódicos viven únicamente en el abstracto mundo de las matemáticas como meras curiosidades sin aplicaciones o ejemplos en el mundo "real", sin embargo en la presente sección revisaremos cómo el concepto de estructuras cuasiperiódicas (y más en general, el de estructuras aperiódicas) surge de estudiar algo tan familiar para la mayoría de nosotros como lo son los rompecabezas y cómo, una vez identificadas, dichas estructuras nos han servido, entre otras cosas, como modelo teórico de los hoy denominados cuasicristales.

2.2.1. Teselados

Desde muy temprano en la historia de la humanidad, el ser humano ha incursionado en el arte a partir de sus diferentes representaciones, siendo los mosaicos una de las más antiguas y de suma relevancia para los fines de esta sección. En el contexto del arte, entendemos por mosaico a toda aquella decoración de una superficie conseguida a partir de trozos de piedras de distintos tamaños y colores (o de otros materiales como

Figura 2.1: Mosaico romano con un diseño geométrico que data de finales del siglo 1 de nuestra era. Imagen recuperada de la "Ancient History Encyclopedia": https://www.ancient.eu/image/1283/roman-geometric-mosaic/.

cerámica o pasta vítrea) denominados teselas, los cuales son dispuestos de tal forma que cubran por completo, sin solaparse ni dejar huecos, a la superficie en cuestión [8] (véase la figura 2.1).

Es a partir de esta técnica artística que nace el término "teselado" como concepto matemático que abstrae las propiedades básicas de los mosaicos, con la única diferencia de que ahora cada una de las teselas es un subconjunto cerrado del espacio finito que se desea cubrir (esta idea puede extenderse a espacios infinitos), respetando la condición de que las teselas, salvo quizás por sus fronteras, sean ajenas entre sí y la unión de todas ellas cubra por completo al espacio de interés (una discusión más a profundidad sobre los teselados matemáticos se presenta en la sección dedicada al teselado de Voronoi dentro del capítulo 4 del presente texto). La variedad de diseños y patrones con los que puede formarse un teselado es infinita, y existen en general dos grandes categorías para clasificarlos: Los teselados periódicos y los teselados no periódicos.

Un teselado periódico se define como aquel en el cual es posible determinar una región del espacio de tal forma que todas las teselas pertenecientes al teselado sean únicamente una traslación de dicha región, sin rotarla ni reflejarla [9]. Particularmente famosos son los teselados periódicos del artista gráfico holandés M. C. Escher, quien en sus diseños empleó teselas que evocaban distintos animales, véase por ejemplo la figura 2.2.

Un teselado no periódico es aquel que no satisface la condición esencial de los teselados periódicos, véase por ejemplo la figura A de la imagen 2.3 en donde se emplean

Figura 2.2: Teselación hexagonal con animales del artista gráfico holandés M. C. Escher: *Study of Regular Division of the Plane with Reptiles* (1939). Imagen recuperada de https://en.wikipedia.org/wiki/M._C._Escher

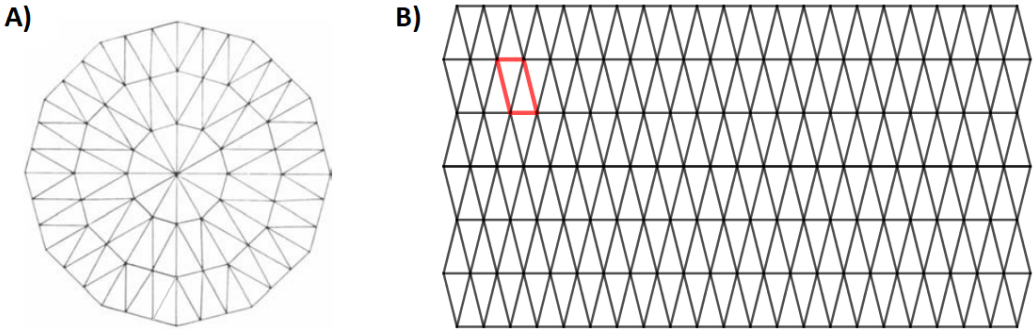

Figura 2.3: A) Teselado no periódico a partir de triángulos isósceles [9]. **B)** Teselado periódico a partir de triángulos isósceles, remarcado en contorno rojo se muestra la tesela correspondiente.

como teselas triángulos isósceles congruentes (esto es, que todos los triángulos tienen las mismas dimensiones y la misma forma). Sin embargo, para todos los casos conocidos de teselados no periódicos formados a partir de teselas congruentes es posible generar un teselado periódico a partir de redefinir las teselas, más no así las formas empleadas; véase por ejemplo la figura B de la imagen 2.3 en donde se siguen empleando triángulos isósceles congruentes, sin embargo la tesela debe redefinirse ahora como la unión de dos de estos triángulos como se señala en dicha figura empleando un contorno rojo.

Bajo este contexto surgió la siguiente pregunta *¿Existe algún conjunto de teselas, con dos o más figuras diferentes, que formen únicamente un teselado no periódico?* La condición de formar únicamente un teselado no periódico implica que, a partir de cualquier subconjunto (incluyendo el conjunto completo) de dichas teselas, no es posible formar un teselado periódico, pero considerando al conjunto completo, estas pueden

formar un teselado no periódico. Para responder a esta pregunta es válido considerar rotaciones y reflexiones. A este conjunto, en caso de existir, se le denomina un **conjunto aperiódico de teselas**.

La respuesta a esta pregunta comenzó a gestarse de la mano de H. Wang en 1961 [10] cuando presenta su investigación sobre la decidibilidad del "problema del teselado", el cual consiste en determinar cuándo un conjunto de prototeselas puede formar un teselado infinito del plano. Para entender mejor esta cuestión es necesario definir qué entendemos por un conjunto de prototeselas, para ello consideremos lo siguiente: Sea $\mathbb{T}$ un teselado cualquiera y sea $\mathbb{P}$ un conjunto de figuras o formas en el plano de tal manera que ninguno de sus elementos es congruente a algún otro elemento de dicho conjunto, con la condición adicional de que cualquier tesela de $\mathbb{T}$ es congruente con algún elemento de $\mathbb{P}$. Al conjunto $\mathbb{P}$ que satisface estas condiciones se le denomina el **conjunto de prototeselas** del teselado $\mathbb{T}$ [11]. En otras palabras, el conjunto de prototeselas de un teselado es el menor conjunto de figuras geométricas distintas a partir de las cuales, haciendo diferentes copias de ellas, se cubre por completo el espacio.

Wang llegó a la conclusión de que el problema es decidible si se niega la existencia de los conjuntos aperiódicos de teselas y conjeturó que dichos conjuntos no existen, sin embargo en 1966, R. Berger [12] demostró que dicho problema es indecidible y fue el primero en mostrar un conjunto aperiódico de teselas. Este resultado llamó el interés de la comunidad matemática, quienes comenzaron a buscar conjuntos aperiódicos de teselas con un conjunto de prototeselas asociado cada vez más pequeño; no fue sino hasta el año 1974 que el físico y matemático Roger Penrose (ganador del Premio Nobel de Física en el 2020) logró encontrar un conjunto aperiódico de teselas cuyo conjunto de prototeselas contiene únicamente dos elementos [9]. Resulta interesante como anécdota histórica el hecho de que, debido a las posibilidades prácticas (y comerciales) de generar rompecabezas basados en el conjunto de prototeselas encontrado por Penrose, este se mostrara renuente a publicar sus descubrimientos hasta no haber patentado su descubrimiento. Hoy, al teselado que se genera con dicho conjunto de prototeselas se le conoce como **el teselado de Penrose** (véase la figura 2.4).

Una de las propiedades más interesantes que presenta el teselado de Penrose (y en general cualquier teselado generado a partir de un conjunto aperiódico de teselas) es la autosimilitud, propiedad que queda de manifiesto al considerar el método de **inflación-deflación** para generar estos teselados (los detalles técnicos de dicho método se exponen en la sección 2.3. dentro del presente capítulo).

2.2.2. Cuasicristales

La teoría clásica sobre las estructuras atómicas de los cristales estaba basada en la presencia de órdenes traslacionales de largo alcance con una distribución periódica de celdas unitarias [13]; sin embargo, en 1984, Levine y Steinhardt [14] (basándose en

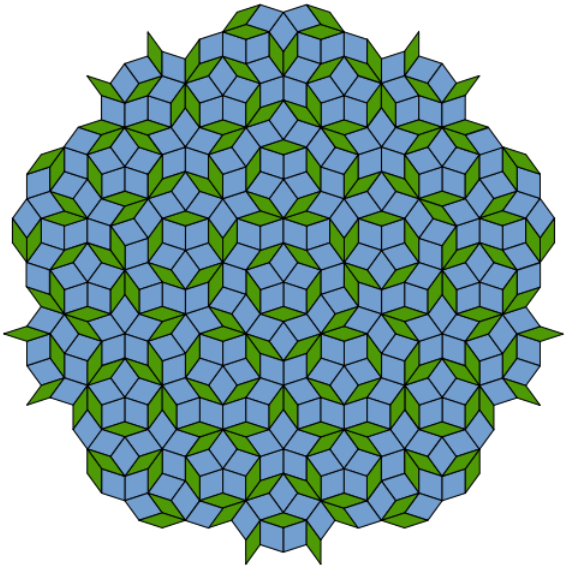

Figura 2.4: Teselación de Penrose en el plano dos dimensional. Señaladas por colores se distinguen las dos figuras que conforman al conjunto de prototeselas. Imagen recuperada de https://es.wikipedia.org/wiki/Teselación_de_Penrose.

los resultados experimentales obtenidos por Shechtman et al. [1] sobre el patrón de difracción de electrones en una aleación de Aluminio y Manganeso (86 % Al, 14 % Mn) al ser rápidamente enfriada) propusieron un nuevo tipo de estructura atómica que denominaron **cuasicristalina**.

Esta nueva estructura atómica genera patrones de difracción discretos (característica atribuida hasta ese entonces a los cristales sólidos) sin poseer simetría traslacional, teniendo en su lugar un parámetro de orden orientacional de largo alcance. Como consecuencia de la ausencia de simetría traslacional en su estructura atómica, los cuasicristales pueden poseer simetrías rotacionales arbitrarias, en particular aquellas asociadas a cualquier polígono regular de N lados [15].

Debido a las características de aperiodicidad y parámetros de orden orientacional de largo alcance que presentan los teselados autosimilares como el teselado de Penrose, los teselados generados a partir de un conjunto aperiódico de teselas fueron reconocidos rápidamente como un modelo matemático apropiado para describir la estructura atómica de los cuasicristales [16].

Tras las publicaciones de Shechtman, Levine y Steinhardt, el número de publicaciones reportando diferentes variedades de cuasicristales fue en aumento, motivo por el cual la Unión Internacional de Cristalografía (IUC por sus siglas en inglés) modificó en 1991 las definiciones de cristal y cuasicristal, quedando de la siguiente manera:

"*A partir de ahora, entenderemos por cristal cualquier sólido cuyo patrón de difracción sea discreto, siendo un cristal aperiódico* [entre ellos los cuasicristales] *cualquier cristal cuya retícula tres dimensional carezca de periodicidad.*" [17]

Cabe destacar que la definición de cuasicristal dada por la IUC habla sobre la retícula tres dimensional en su totalidad, por lo cual es posible que un cuasicristal presente simetría traslacional en alguna de sus dimensiones (véase por ejemplo [18]), es decir, que la retícula atómica del cuasicristal sea periódica en una o dos dimensiones.

2.2.3. Cuasicristales suaves

Como suele ocurrir cada que alguien nos señala algún detalle que hasta entonces era ignorado por nosotros, momento a partir del cual dicho detalle se hace evidente en diversos contextos, una vez que se aceptó la presencia de estructuras cuasiperiódicas en las diferentes aleaciones metálicas empleadas para generar cuasicristales, estas comenzaron a ser observadas y estudiadas en diversas áreas.

En el 2004 Zeng et al. [2] reportaron una estructura con una simetría rotacional dodecagonal en uno de sus planos (siendo periódica en el eje ortogonal a dicho plano) al estudiar el compuesto $[3,4,5-(3,5)^2]$ $12G_3CH_2OH$, el cual es un sistema conformado por dendrones (moléculas en forma de árbol) al que catalogaron como un cuasicristal líquido. Este ejemplo forma parte de los denominados "cuasicristales suaves", estructuras cuasiperiódicas presentes en los sistemas que son objeto de estudio de la materia condensada blanda (o soft matter) tales como líquidos, coloides, espumas, geles, etc.

Una de las características más relevantes sobre los cuasicristales suaves es la escala de longitud que presentan sus estructuras cuasiperiódicas. La longitud de las celdas unitarias de las estructuras cuasiperiódicas asociadas a los cuasicristales formados por aleaciones metálicas es del orden de 0.5 nm, siendo del orden de 2 nm para calcogenoides, 10 nm para cuasicristales líquidos formados por supramoléculas dendrímeras (con forma de árbol) así como para sílices mesoporosas, 20 nm para nanopartículas binarias, 50 nm para coloides y entre 50 y 80 nm para polímetros lineales [3].

En el 2011, Fischer et al. [4] reportaron la (auto)formación de estructuras cuasiperiódicas con simetrías rotacionales dodecagonal y octodecagonal en coloides cuya fase continua fue agua y cuya fase dispersa consistió en poli(óxido de isopreno-b-etileno), PI_n-PEO_m, con n y m los diferentes grados de polimerización de los polímeros; dichas estructuras se observaron para los casos particulares $PI_{30}-PEO_{120}$, $PI_{30}-PEO_{124}$ y $PI_{32}-PEO_{120}$ a diferentes temperaturas y concentraciones. Este fue el primer caso en el que se observó una estructura cuasiperiódica octodecagonal.

Actualmente se siguen estudiando diversos aspectos sobre los cuasicristales suaves,

entre los que destacan un análisis sobre su estabilidad termodinámica [19], nuevos modelos computacionales para simular el crecimiento de cuasicristales bidimensionales, tanto en coloides [20], como estructuras moleculares [21] o simulaciones empleando métodos de Monte Carlo para estudiar, a partir de un potencial de Lennard-Jones-Gauss con dos mínimos locales, la transición de líquido a sólido en cuasicristales bidimensionales con simetría decagonal en coloides [22].

2.2.4. Cuasicristales fotónicos

Los cuasicristales fotónicos son un tipo de materiales ópticos los cuales forman parte de los denominados "materiales con brecha de banda fotónica" (o PBG por sus siglas en inglés), estos se caracterizan por ser capaces de prohibir la propagación de la luz, para ciertos intervalos de frecuencia, en una o más direcciones [23]. Esta capacidad de manipular la transmisión de la luz resulta bastante atractiva para su aplicación en la comunicación óptica [24], la creación de dispositivos de filtrado óptico [25, 26], la fabricación de láseres [27], el encubrimiento óptico [28], entre otros. Los detalles técnicos acerca de la existencia de estas brechas fotónicas en sistemas cuasiperiódicos pueden encontrarse en [23, 29].

A principios del 2020, Jin et al. [30] describieron un método experimental para generar cuasicristales fotónicos de simetría octogonal en dos dimensiones. Este método conocido como **método por interferencia de láseres** se basa en el efecto fotorefractivo presente en algunos materiales, el cual se manifiesta como una variación del índice de refracción del material a causa de una fuente luminosa que incide sobre él. De esta manera, al emplear múltiples láseres coherentes, se pueden formar patrones periódicos o cuasiperiódicos en el índice de refracción de un material fotorefractivo, dando lugar a un cuasicristal fotónico.

Este método es de particular interés para los fines de este texto debido a que en principio permite la formación de estructuras cuasiperiódicas de cualquier simetría, lo cual nos proporciona un método experimental para corroborar los resultados obtenidos con el algoritmo computacional que se expone en la presente tesis.

Otro método similar al anterior consiste en aplicar la interferencia de láseres a sustratos coloidales para forzar a las partículas presentes en el coloide a adquirir una estructura cuasiperiódica mediante trampas ópticas [31].

2.3. Métodos de construcción

A partir de su descubrimiento se han desarrollado varios modelos para estudiar las estructuras cuasiperiódicas subyacentes a los cuasicristales, siendo estos principalmente la descripción "density-wave" que conduce a una teoría de Landau para cuasicristales,

así como la descripción basada en celdas unitarias que describe a dichas estructuras como distribuciones cuasiperiódicas de dos o más celdas unitarias distintas [32] (es decir, los teselados generados a partir de un conjunto aperiódico de teselas presentados en la sección anterior).

Actualmente existen varios algoritmos para generar estructuras cuasiperiódicas de manera computacional, entre los que destacan el método de inflación-deflación, los métodos de corte y proyección y el método dual generalizado (o GDM por sus siglas en inglés). A continuación expondremos brevemente en qué consiste cada uno de dichos métodos.

2.3.1. Método de inflación-deflación

El contenido de los siguientes párrafos en los que se desarrolla la parte teórica necesaria para describir el método de inflación-deflación corresponden al artículo de Frank [16], el cuál es una introducción a la teoría de las reglas de sustitución para los teselados en el plano euclídeo; el lector interesado puede revisar dicho artículo para una discusión más detallada del tema.

El método de inflación-deflación consiste, en términos generales, en definir una serie de reglas de sustitución para cada una de las prototeselas de algún teselado $\mathbb{T}$ que estemos interesados en generar, de tal forma que tras aplicar la correspondiente regla de sustitución a alguna tesela $t \in \mathbb{T}$, el resultado sea una región del espacio que mantiene la misma forma geométrica que la tesela t pero conformada por la unión de varias teselas. De esta forma, dado un conjunto inicial de teselas y aplicando de manera iterada las reglas de sustitución a cada una de las teselas que se encuentran en el espacio, es posible ir generando el teselado completo.

Para poder establecer de manera formal esta idea, precisamos definir los siguientes términos:

- **Definición 2.3.1.1:** Sea $\mathbb{T}$ un teselado arbitrario con $t, t^{'} \in \mathbb{T}$ dos teselas. Diremos que t y $t^{'}$ son **teselas equivalentes** si ellas difieren únicamente por un movimiento rígido.
- **Definición 2.3.1.2:** A la unión finita de teselas cuya intersección sea vacía (salvo por sus fronteras) y que cubran una región conexa del espacio le llamaremos **parche**.
- **Definición 2.3.1.3:** Sea $\mathbb{T}$ un teselado arbitrario con P y $P^{'}$ dos parches de dicho teselado. Diremos que P y $P^{'}$ son **parches equivalentes** si existe un movimiento rígido entre ellos de modo que las teselas que se correspondan al hacer dicho movimiento rígido sean equivalentes.

- **Definición 2.3.1.4:** Un teselado $\mathbb{T}$ se dice que tiene **complejidad local finita** si el número de parches de $\mathbb{T}$ conformados por dos teselas es finito, salvo equivalencias.

- **Definición 2.3.1.5:** Un teselado $\mathbb{T}$ se denomina **repetitivo** si para cada parche P de $\mathbb{T}$ existe un número real $R > 0$ tal que, dentro de toda bola de radio R en el espacio, existe un parche $P^{'}$ que es equivalente al parche P.

Una vez presentadas las definiciones previas, procedemos a analizar las características esenciales que permiten construir a dichos teselados a partir de este método:

Sea $\phi : \mathbb{R}^d \to \mathbb{R}^d$ una transformación lineal que se expande, en el sentido de que el módulo de todos sus eigenvalores es mayor a uno (esta transformación lineal ϕ juega el papel de la regla de sustitución para las prototeselas). Un teselado $\mathbb{T}$ es *ϕ-dividido* si satisface que:

1. Para cada tesela $t \in \mathbb{T}$, $\phi(t)$ es un parche de $\mathbb{T}$
2. t y $t^{'}$ son dos teselas equivalentes si y sólo si los parches generados por $\phi(t)$ y $\phi(t^{'})$ son equivalentes.

Un teselado $\mathbb{T}$ se denomina **auto-afín con mapeo de expansión** ϕ si es ϕ-dividido, repetitivo y tiene una complejidad local finita. Si además, el mapeo lineal ϕ es tal que preserva los ángulos y las distancias, el teselado $\mathbb{T}$ se denomina **auto-similar**.

A la regla de sustitución que toma una tesela $t \in \mathbb{T}$ y regresa una unión de teselas a partir del mapeo $\phi(t)$ se le conoce como el **método de inflación-deflación** dado que primero "infla" una tesela a través del mapeo ϕ y después descompone la imagen en la unión de varias teselas en la escala original.

Para entender mejor lo descrito en párrafos anteriores, veamos el siguiente ejemplo:

- **Teselado L-Triominó:** Considérense las cuatro figuras en forma de "L" que se muestran a la izquierda de cada flecha negra en la figura 2.5, cada una de ellas corresponde a una prototesela diferente del teselado que buscamos construir. A la derecha de dichas flechas se encuentra el resultado de aplicar la regla de sustitución o mapeo ϕ sobre cada una de las prototeselas; nótese como el mapeo produce un parche conformado por cuatro teselas que mantiene la forma geométrica y orientación de la prototesela sobre la cual dicho mapeo es aplicado.

  De esta manera, comenzando con una única tesela amarilla, podemos ir iterando la regla de sustitución para generar el teselado correspondiente, véase la figura 2.6.

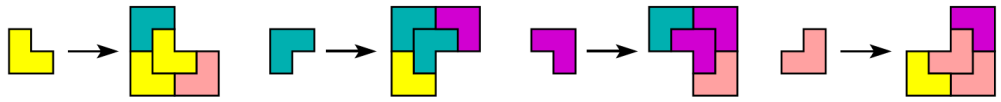

Figura 2.5: Prototeselas y su correspondiente regla de sustitución del teselado L-Trominó. Imagen recuperada de [16]

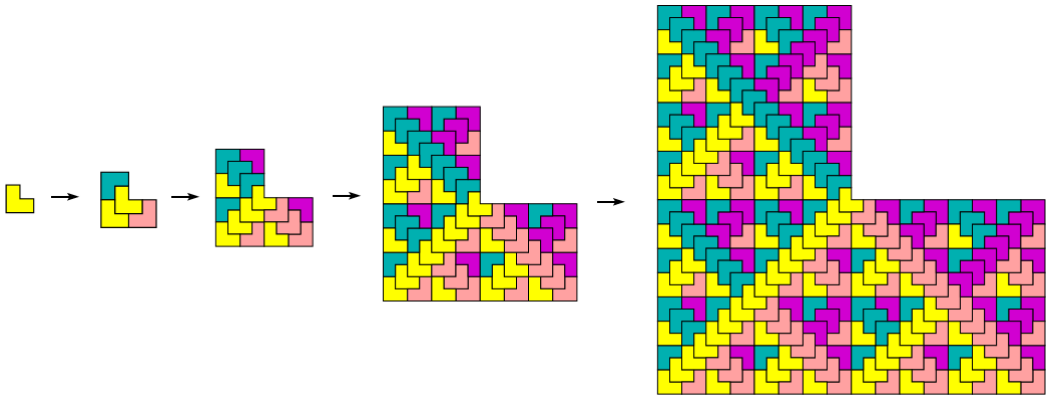

Figura 2.6: Primeras cuatro iteraciones de la regla de sustitución para el teselado L-Trominó. Imagen recuperada de [16]

Las condiciones expuestas previamente referentes a la equivalencia entre teselas pueden ser relajadas para permitir rotaciones además de las traslaciones, lo cual abre el abanico de teselados que pueden generarse a partir de este método.

Para el lector interesado en cómo generar el famoso teselado de Penrose a partir del método de inflación-deflación, en el artículo de Gardner [9] se presentan las prototeselas a emplear así como los detalles sobre la regla de sustitución; un acercamiento ligeramente diferente para este mismo objetivo puede encontrarse en el artículo de Frank [16] en donde define un par de prototeselas diferente y una variación del método de inflación-deflación en la que se permite que la regla de sustitución no preserve exactamente la forma de la tesela sobre la cual actúa.

2.3.2. Método de corte y proyección

La presente subsección fue presentada originalmente en mi tesis de licenciatura [33] donde presenté un algoritmo computacional basado en un método de corte y proyección para estudiar la difusión de un sistema cuasiperiódico, me he tomado la libertad de incluirlo nuevamente por motivos de completez.

El contenido de los siguientes párrafos en los que se desarrolla parte de la teoría matemática en la que se basan los métodos de corte y proyección corresponden al libro

"Quasicrystals and geometry" del autor Marjorie Senechal [34], el lector interesado en profundizar en el tema puede referirse a dicha fuente.

La idea base para generar retículas cuasiperiódicas bajo este enfoque requiere pensar en dimensiones mayores a la de la retícula que queremos obtener debido a la definición de funciones cuasiperiódicas expuesta previamente, las cuales están formadas por más de una función periódica, con lo cual podemos interpretar a la dimensión de nuestra red[1] cuasiperiódica como el número de funciones periódicas que requerimos para definir la función cuasiperiódica que generará a nuestra red.

Para el caso más sencillo posible pensemos en la retícula bidimensional cuadrada y una partícula desplazándose en línea recta sobre el plano que la contiene. Debido a que la separación de los puntos que conforman esta lattice es igual en ambos ejes coordenados, la periodicidad del movimiento de la partícula sobre el plano recae únicamente en las proyecciones de la velocidad sobre los ejes coordenados. De esta manera, si las proyecciones de la velocidad son inconmensurables entre si, es decir que la pendiente de la recta que sigue la partícula es irracional, podemos pensar en el movimiento que describe la partícula con respecto al vértice más cercano en la retícula como la suma de dos movimientos periódicos con periodos inconmensurables entre sí, es decir, una función cuasiperiódica.

Notemos que hasta ahora, esto nos genera una función cuasiperiódica continua, sin embargo queremos generar una retícula, por lo cual estamos interesados en funciones cuasiperiódicas discretas. Con la función generada hasta ahora podemos seleccionar un subconjunto de puntos de dicha función, correspondiente a los mínimos locales de la función, este conjunto corresponderá a los vértices de nuestra red cuasiperiódica unidimensional. El conjunto de puntos formado al tomar los mínimos locales de nuestra función cuasiperiódica continua es equivalente a considerar la proyección de los vértices más cercanos a la línea que describe la partícula al desplazarse.

Este método de selección de los vértices de la red que proyectaremos a la trayectoria de la partícula es equivalente a considerar a los vértices que conforman a los centros de los polígonos de Voronoi formados a partir de la retícula periódica dos dimensional por los cuales pasa la trayectoria de la partícula. Esto a su vez es equivalente [35] a considerar aquellos vértices de la retícula periódica bidimensional que caen dentro de una banda formada por el producto cruz de la trayectoria y la proyección de un cuadrado de lado uno centrado en el origen sobre el espacio ortogonal a la trayectoria de la partícula.

Podríamos no sólo haber tomado la proyección de los vértices más cercanos, sino también considerar a los vértices más cercanos y a los segundos más cercanos o, incluso, generalizar esta línea de pensamiento a todos aquellos vértices que se encuentren dentro de una banda determinada formada por el producto cruz de la trayectoria y cualquier otra figura en el espacio ortogonal a dicha trayectoria.

[1] Se utilizará como sinónimos a lo largo del texto las palabras retícula, red y lattice.

Todo lo anterior es la motivación del método de corte y proyección, sin embargo nótese que hasta ahora hemos empleado términos como retícula, proyección ortogonal o discreto basándonos en la idea intuitiva de dichos conceptos. Procedemos ahora a definirlos de manera formal.

- **Definición 2.3.2.1:** Sea G un grupo de transformaciones actuando sobre un "objeto matemático" M. (M puede ser todo el espacio euclidiano de n dimensiones (E^n), un objeto geométrico como un cubo o incluso G mismo). Sea x un punto de M.

  La órbita de x bajo G se define como el conjunto:

$$O_G(x) = \{gx \in M | g \in G\}$$

- **Definición 2.3.2.2:** Sean $\vec{b}_1, \vec{b}_2, ..., \vec{b}_k \in E^n$ un conjunto de vectores. Al grupo infinito numerable generado por estos vectores bajo la suma usual, de modo que los elementos del grupo sean de la forma $\vec{x} = m_1\vec{b}_1 + m_2\vec{b}_2 + ... + m_k\vec{b}_k$ con $m_i \in \mathbb{Z}$ se le denomina módulo Z.

- **Definición 2.3.2.3:** Sea Ω la órbita de un módulo Z generado por los vectores $\vec{b}_1, \vec{b}_2, ..., \vec{b}_k \in E^n$. Si el rango de Ω es igual a la dimensión del subespacio generado por los vectores $\vec{b}_1, \vec{b}_2, ..., \vec{b}_k$ decimos que Ω es una retícula.

En lo subsiguiente denotaremos por "retícula puntual" ($\mathbb{L}^p$) a una órbita generada por un módulo Z cuyos vectores generadores sean linealmente independientes, mientras que usaremos el término "retícula" ($\mathbb{L}$) para referirnos al módulo Z.

- **Definición 2.3.2.4:** Sean $\vec{b}_1, \vec{b}_2, ..., \vec{b}_k$ una base para $\mathbb{L}$. Definimos a la norma $N(\vec{b}_i)$ de $\vec{b}_i$ como:

$$N(\vec{b}_i) = |\vec{b}_i|^2 = \vec{b}_i \cdot \vec{b}_i,$$

  con $\cdot$ el producto punto usual.

- **Definición 2.3.2.5:** Decimos que $\mathbb{L}$ es una retícula entera si, para todo $\vec{x} \in \mathbb{L}$, $N(\vec{x}) \in \mathbb{Z}$.

- **Definición 2.3.2.6:** Sea $\mathbb{L}$ una retícula entera y ε cualquier subespacio k-dimensional de E^n donde $0 < k < n$. Si $\varepsilon \cap \mathbb{L} = \{0\}$ entonces se dice que ε es totalmente irracional.

- **Definición 2.3.2.7:** Sea P_L un mapeo de un espacio de Hilbert H a un subespacio L de H. Si el mapeo es tal que $x - P_L(x)$ es ortogonal a $P_L(x)$ (es decir, $x - P_L(x) \perp P_L(x)$) con $x \in H$, se dice que el mapeo P_L es un proyector ortogonal de H a L [36].

- **Definición 2.3.2.8:** Sea $\Lambda \subset E^n$ un conjunto de puntos. Se dice que Λ es discreto si existe algún número real $r_0 > 0$ tal que para todo $x, y \in \Lambda$ se cumple $|x - y| \geq 2r_0$.

- **Definición 2.3.2.9:** Un conjunto de puntos Λ es relativamente denso en E^n si existe algún número real R_0 tal que toda esfera de radio mayor que R_0 en E^n contiene al menos un punto de Λ en su interior.

- **Definición 2.3.2.10:** Un conjunto de puntos $\Lambda \subset E^n$ tal que sea discreto y relativamente denso en E^n se denomina como conjunto de Delone.

Sea ε un subespacio d-dimensional totalmente irracional de E^n y sea $\varepsilon^\perp$ su complemento ortogonal ($\varepsilon^\perp$ puede o no ser totalmente irracional). Sea Π el proyector ortogonal a ε y $\Pi^\perp$ el proyector ortogonal a $\varepsilon^\perp$. Dado que Π y $\Pi^\perp$ son mapeos lineales, entonces $\Pi(\mathbb{L}^p)$ y $\Pi^\perp(\mathbb{L}^p)$ son órbitas de los módulos Z en ε y $\varepsilon^\perp$ respectivamente, generados por las proyecciones de cualquier base para $\mathbb{L}^p$ en dichos subespacios.

Dado que el conjunto de puntos $\Pi(\mathbb{L}^p)$ es denso en ε, $\Pi(\mathbb{L}^p)$ no es un conjunto de Delone (los conjuntos en los que estamos interesados para modelar a los cuasicristales son precisamente conjuntos de Delone). Para obtener uno que sí lo sea debemos seleccionar un subconjunto de puntos de $\mathbb{L}^p$ para proyectarlos sobre ε.

Sea $K \subset \varepsilon^\perp$ un subconjunto compacto tal que $K \cap \mathbb{L}^p \neq \phi$, llamaremos a K el dominio de aceptación para la proyección. El subconjunto de puntos de $\mathbb{L}^p$ que proyectaremos sobre ε serán aquellos puntos $x \in \mathbb{L}^p$ tales que $\Pi^\perp(x) \in K$. Estos puntos son aquellos que caen dentro del cilindro $C = K \oplus \varepsilon$ (Véase la figura 2.7).

- **Proposición 2.3.2.1:** Sea $X = C \cap \mathbb{L}^p$. El conjunto de puntos $\Pi(X)$ es un conjunto de Delone.

  Demostración: Debemos mostrar que el conjunto $\Pi(X)$ es discreto y relativamente denso en ε. Para probar el que sea discreto basta con mostrar que hay una vecindad alrededor del origen en ε que no contiene ningún otro punto de $\Pi(X)$ más que el origen. Sea $x \in X$ y supongamos que $\Pi(x)$ cae en $B_0(c)$, la bola de radio $c > 0$ alrededor del origen. Dado que

  $$|x|^2 = |\Pi(x)|^2 + |\Pi^\perp(x)|^2$$

  y que el conjunto $\Pi^\perp(X)$ es un conjunto acotado por hipótesis, $\Pi(x)$ debe caer en una esfera de radio finito $m > 0$ alrededor del origen. Dado que $\mathbb{L}^p$ es discreta (pues estamos suponiendo que nuestra retícula $\mathbb{L}$ es entera) entonces $\mathbb{L}^p \cap B_0(m)$ es un subconjunto finito $U \subset \mathbb{L}^p$. Por lo anterior, el conjunto de puntos $\Pi(U)$ es finito y para algún valor $r > 0$ tenemos que $\Pi(U) \cap B_0(r) = \{0\}$.

  La densidad relativa es inmediata debido a que $\mathbb{L}^p$ es relativamente densa en E^n y por lo tanto en el cilindro C.

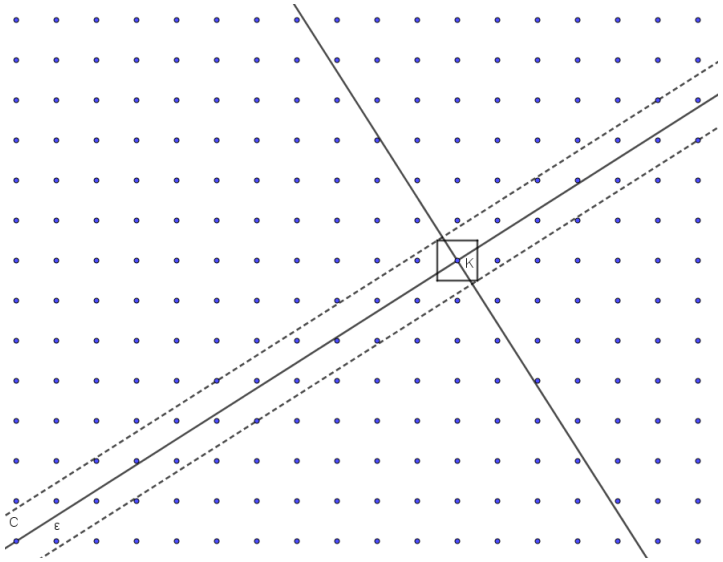

Figura 2.7: Ejemplo del cilindro C determinado por el dominio de aceptación K para una proyección de una retícula integral en dos dimensiones a un subespacio ε totalmente irracional de una dimensión . Imagen creada en el software **Paint**.

- **Proposición 2.3.1.2:** Si ε es totalmente irracional, entonces $\Pi(X)$ es no periódico.

  Demostración: Descompongamos a $\varepsilon^{\perp}$ en los subespacios V y W tal que $\varepsilon^{\perp} = V \oplus W$. Dado que $\Pi^{\perp}(\mathbb{L})$ es uno a uno y la dimensión de $\varepsilon^{\perp} < n$, el subespacio V es no trivial y la restricción de Π a $(V \oplus \varepsilon) \cap \mathbb{L}$ es uno a uno puesto que $\Pi^{\perp}((V \oplus \varepsilon) \cap \mathbb{L})$ es denso en V. Entonces, dado que $K \cap \mathbb{L}^p \neq \phi$, $(K \cap V) \cap \Pi^{\perp}(\mathbb{L})$ es denso en $K \cap V$. Definamos $X_V = ((K \cap V) \oplus \varepsilon) \cap \mathbb{L}$. Si $\Pi(X_V)$ es periódico entonces X_V lo es. Sin embargo, X_V no puede ser periódico pues K es compacto y por lo tanto no puede contener todos los múltiplos enteros de cualquier vector. Así, siendo que $(K \cap W) \cap \Pi^{\perp}(\mathbb{L})$ es finito, entonces X y por lo tanto $\Pi(X)$ no puede ser periódico.

De esta forma se tiene un método relativamente sencillo de generar retículas k dimensionales cuyo conjuntos de puntos son un conjunto de Delone no periódicos al proyectar subconjuntos de otras retículas enteras en n dimensiones con $k < n$.

2.3.3. Método dual generalizado

El método dual generalizado surge como una generalización al método desarrollado por De Bruijn para obtener teselados tipo Penrose en dos dimensiones [37] y permite generar retículas cuasiperiódicas en dos y tres dimensiones para cualquier simetría rotacional deseada. A grandes rasgos el método dual generalizado consiste en lo siguiente [15, 38]:

- Se genera un conjunto de vectores $\mathbf{V}_s = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N \mid \mathbf{e}_i \in \mathbb{R}^m,\ N \in \mathbf{Z},\ m \in \{2, 3\}\}$ el cual determina la simetría orientacional del arreglo cuasiperiódico deseado. Por ejemplo, para retículas cuasiperiódicas con simetría rotacional asociada a un polígono regular de N lados, los vectores $\mathbf{e}_i$ están dados por

$$\mathbf{e}_i = \left(\cos\left(\frac{2\pi i}{N}\right), \sin\left(\frac{2\pi i}{N}\right)\right),$$

  a este conjunto de vectores $\mathbf{V}_s$ lo denominamos el conjunto de vectores estrella.

- Se genera el conjunto infinito de planos orotogonales a cada uno de los vectores estrella $\mathbf{e}_i$ (rectas ortogonales si consideramos el caso 2D). Estos planos (rectas) pueden estar separados entre sí una distancia constante o siguiendo una secuencia cuasiperiódica. Dicho conjunto es lo que se conoce como un N-mallado y lo denotamos por $\mathbf{G}_N$, de tal forma que su expresión matemática está dada por

$$\mathbf{G}_N = \{\mathbf{x}\ \mathbb{R}^m | \mathbf{x} \cdot \mathbf{e}_i = n_i + \alpha_i + \xi_{n_i};\ i \in \{1, 2, \ldots, N\},\ m \in \{2, 3\},\ n_i \in \mathbb{Z}\}, \quad (2.2)$$

  donde $\alpha_i \in \mathbb{R}$ es un desplazamiento respecto al origen para los planos (rectas) ortogonales al vector $\mathbf{e}_i$ y $\xi_{n_i} \in \mathbb{R}$ define la separación entre un plano (recta) y otro. El caso en que los planos (rectas) están separados entre sí una distancia constante se obtiene al hacer $\xi_{n_i} = 0\ \forall\ n_i$.

- Los planos (rectas) dividen al espacio (plano) en regiones abiertas ajenas a través de las cuales no pasa ningún plano (recta). Cada una de estas regiones está especificada de manera única por la eneada de números enteros $(k_1, k_2, \ldots, k_N)$, estos números enteros se obtienen de la siguiente manera: Sea $\mathbf{x}_0$ un punto arbitrario dentro de una región fija, entonces el entero k_i corresponde al número entero n_i que aparece en la expresión 2.2 tal que el punto $\mathbf{x}_0$ se encuentra entre los planos (rectas) ortogonales al vector $\mathbf{e}_i$ generados al emplear los enteros n_i y $n_i + 1$ en dicha expresión.

- Los puntos que conforman a la retícula cuasiperiódica deseada se construyen mapeando cada una de las regiones abiertas del punto anterior a través del siguiente mapeo

$$\mathbf{t} = \sum_{i=1}^{N} k_i \mathbf{e}_i, \quad (2.3)$$

  donde $\mathbf{t}$ es un punto en $\mathbb{R}^3$ ($\mathbb{R}^2$). Estos puntos corresponden a los vértices de las celdas unitarias que conforman al arreglo cuasiperiódico en la descripción con celdas unitarias de los cuasicristales.

Un análisis más a profundidad acerca de este método y su implementación en algoritmos computacionales se aborda en el siguiente capítulo.

Capítulo 3

Método Dual Generalizado Descentralizado

El método dual generalizado nos proporciona un algoritmo directo para generar una amplia variedad de arreglos cuasiperiódicos de dimensión arbitraria m; a grandes rasgos el algoritmo consiste en dividir el espacio $\mathbb{R}^m$ en un conjunto de regiones abiertas y ajenas entre sí, cada una de las cuales será mapeada a un sitio del arreglo cuasiperiódico deseado.

Los detalles acerca del método dual generalizado pueden ser consultados en el capítulo 2 del presente documento, a partir de los cuales se puede notar que no resulta sencillo obtener, implementando directamente el método, los sitios de un arreglo cuasiperiódico alrededor de un punto arbitrario en el espacio ya que dicho método no nos indica cuál es la relación espacial entre una región abierta del teselado de $\mathbb{R}^m$ y el sitio del arreglo cuasiperiódico al cual dicha región es mapeada.

En el presente capítulo desarrollaremos un algoritmo basado en el método dual generalizado con el cual podemos obtener los sitios, alrededor de un punto arbitrario en el plano, de cualquier arreglo cuasiperiódico bidimensional con simetría rotacional N generado por un mallado con separación periódica.

3.1. Expresiones analíticas para las coordenadas de los sitios de un arreglo cuasiperiódico

Uno de los resultados principales en los cuales se sustenta nuestro trabajo es el algoritmo presentado por Naumis y Aragón [38] para obtener la expresión analítica de las coordenadas de los sitios de un arreglo cuasiperiódico, en dos y tres dimensiones, a partir del método dual generalizado. En el artículo publicado aparece un pequeño

error en la ecuación 6 (los términos deberían estar restándose, no sumándose) el cual se propaga a las ecuaciones 7 y 8, motivo por el cual a continuación se desarrolla el álgebra descrita por los autores para el caso bidimensional, considerando mallados con una separación periódica entre sus rectas.

Dado un conjunto de vectores $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N \mid \mathbf{e}_i \in \mathbb{R}^2,\ N \in \mathbf{Z}^+,\ N \geq 3\}$ que determinan la simetría orientacional del arreglo cuasiperiódico (en lo subsiguiente "vectores estrella"), la expresión general para un mallado G_N formado por dicho conjunto de vectores es

$$G_N = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{x} \cdot \mathbf{e}_i = n_i + \alpha_i;\ i = 1, 2, \ldots, N;\ n_i \in \mathbb{Z}\}, \tag{3.1}$$

donde $\alpha_i \in (0, 1)$ es el desplazamiento respecto al origen del conjunto de rectas ortogonales asociadas al vector e_i.

Sea $\mathbf{x}$ el punto de intersección de únicamente dos rectas arbitrarias del mallado G_N. Las coordenadas de dicho punto están determinadas por la solución del siguiente sistema de ecuaciones,

$$\mathbf{x} \cdot \mathbf{e}_j = n_j + \alpha_j, \tag{3.2}$$

$$\mathbf{x} \cdot \mathbf{e}_k = n_k + \alpha_k, \tag{3.3}$$

para algunos índices j y k. Dicho sistema puede ser reescrito en forma matricial como

$$\begin{pmatrix} e_{jx} & e_{jy} \\ e_{kx} & e_{ky} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} n_j + \alpha_j \\ n_k + \alpha_k \end{pmatrix},$$

resolviendo para el vector (x, y) tenemos

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{e_{jx}e_{ky} - e_{jy}e_{kx}} \begin{pmatrix} e_{ky} & -e_{jy} \\ -e_{kx} & e_{jx} \end{pmatrix} \begin{pmatrix} n_j + \alpha_j \\ n_k + \alpha_k \end{pmatrix}$$

$$= \frac{1}{A_{jk}} \begin{pmatrix} e_{ky}\,(n_j + \alpha_j) - e_{jy}\,(n_k + \alpha_k) \\ -e_{kx}\,(n_j + \alpha_j) + e_{jx}\,(n_k + \alpha_k) \end{pmatrix},$$

donde $A_{jk} = e_{jx}e_{ky} - e_{jy}e_{kx}$ es el área del rombo generado por los vectores $\mathbf{e}_j$ y $\mathbf{e}_k$.

Dado el vector $\mathbf{e}_i = (e_{ix}, e_{iy})$ definimos a su vector ortogonal como $\mathbf{e}_i^{\perp} = (e_{iy}, -e_{ix})$. Empleando esta notación podemos expresar al punto de intersección $\mathbf{x}$ como

$$\mathbf{x} = \frac{1}{A_{jk}} \left[(n_j + \alpha_j)\,\mathbf{e}_k^{\perp} - (n_k + \alpha_k)\,\mathbf{e}_j^{\perp} \right]. \tag{3.4}$$

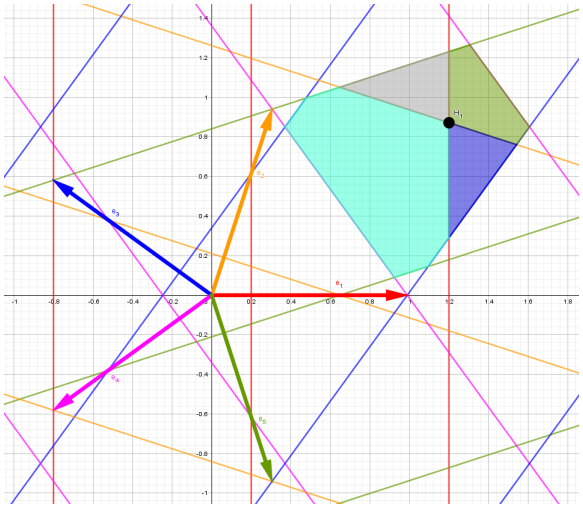

Figura 3.1: Mallado G_N generado por un conjunto de vectores estrella correspondiente a los vértices de un pentágono regular y constantes $\alpha_i = 0.2$ para todo valor de i. El punto negro corresponde a la intersección de la recta determinada por $\mathbf{x}_1 \cdot \mathbf{e}_1 = 1.2$ y la recta determinada por $\mathbf{x}_2 \cdot \mathbf{e}_2 = 1.2$; en colores sólidos las cuatro regiones abiertas determinadas por dicho punto.

Bajo el supuesto de que por el punto $\mathbf{x}$ pasan únicamente dos rectas, se tiene que alrededor de dicho punto existen cuatro regiones abiertas delimitadas por las rectas que conforman a G_N (véase la figura 3.1).

Los sitios que conforman al arreglo cuasiperiódico corresponden al mapeo realizado por la transformación dual que manda a una de estas cuatro regiones al punto dado por

$$t^0_{n_j,n_k} = \sum_{i=1}^{N} n_i \mathbf{e}_i, \tag{3.5}$$

donde $n_i = \min(a, b)$ con a y b los números enteros que generan las rectas ortogonales al vector $\mathbf{e}_i$ entre las cuales está comprendida la región en cuestión.

Estos números enteros n_i se obtienen al proyectar el punto de intersección $\mathbf{x}$ con el respectivo vector estrella $\mathbf{e}_i$, restándole el parámetro α_i y tomando el entero más cercano por debajo de dicho valor. Expandiendo la ecuación 3.5 y sustituyendo los números enteros n_i por su correspondiente expresión, tenemos que el sitio del arreglo cuasiperiódico $t^0_{n_j,n_k}$ está dado por

$$t^0_{n_j,n_k} = n_j \mathbf{e}_j + n_k \mathbf{e}_k + \sum_{i \neq j \neq k}^{N} \left\lfloor \frac{1}{A_{jk}} \left[(n_j + \alpha_j)\, \mathbf{e}_k^{\perp} - (n_k + \alpha_k)\, \mathbf{e}_j^{\perp} \right] \cdot \mathbf{e}_i - \alpha_i \right\rfloor \mathbf{e}_i, \tag{3.6}$$

donde $\lfloor \cdot \rfloor$ es la función piso. Los otros tres sitios asociados a las otras tres regiones están dados por

$$t^1_{n_j,n_k} = t^0_{n_j,n_k} - \mathbf{e}_j, \tag{3.7}$$

$$t^2_{n_j,n_k} = t^0_{n_j,n_k} - \mathbf{e}_j - \mathbf{e}_k, \tag{3.8}$$

$$t^3_{n_j,n_k} = t^0_{n_j,n_k} - \mathbf{e}_k. \tag{3.9}$$

3.2. Descentralizando el método dual generalizado

En lo subsiguiente nos centraremos únicamente en arreglos cuasiperiódicos con simetría rotacional N, es decir aquellos generados a partir de un conjunto de vectores estrella $\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}$ donde $\mathbf{e}_i = \left(\cos\left(\frac{2(i-1)\pi}{N}\right), \sin\left(\frac{2(i-1)\pi}{N}\right)\right)$ y que cumplan la condición de que $\alpha_i = \alpha, \forall\ i \in \{1, \ldots, N\}$.

Las expresiones 3.6 - 3.9 nos permiten generar los vértices de alguno de los polígonos que conforman al arreglo cuasiperiódico conociendo los números enteros n_j y n_k asociados a algún par de vectores estrella $\mathbf{e}_j$ y $\mathbf{e}_k$; sin embargo, este resultado no nos brinda un método con el cual, para algún punto en el plano, podamos encontrar las combinaciones de vectores estrella (así como sus números enteros asociados) que generen los sitios del arreglo cuasiperiódico que se encuentran alrededor de nuestro punto de interés.

En otras palabras, las expresiones 3.6 - 3.9 permiten generar arreglos cuasiperiódicos de simetría rotacional arbitraria alrededor del origen de manera muy sencilla a través de un algoritmo computacional que considere todas las posibles parejas válidas de vectores estrella $(\mathbf{e}_j, \mathbf{e}_k)$ (esto es, que no sean colineales) y que, para cada pareja de vectores, considere todas las posibles parejas de números enteros (n_j, n_k), donde $n_j, n_k \in [c, d]$ con $c, d \in \mathbb{Z}$ tales que $c < 0 < d$. Véase la figura 3.2.

Sin embargo, si estamos interesados en generar una sección del arreglo cuasiperiódico centrada en algún punto fuera del origen, necesitamos encontrar las combinaciones de vectores estrella $(\mathbf{e}_j, \mathbf{e}_k)$ junto con sus correspondientes números enteros (n_j, n_k) que generen los polígonos cercanos al punto en cuestión. A continuación mostramos el algoritmo que hemos desarrollado para obtener dichas combinaciones.

Lo primero que debemos notar es el hecho de que los polígonos del arreglo cuasiperiódico generados al fijar uno de los vectores estrellas de las parejas $(\mathbf{e}_j, \mathbf{e}_k)$ se agrupan formando "franjas ortogonales" al vector estrella fijado. Véase la figura 3.3.

Aproximando cada una de estas "franjas" por la línea recta que mejor ajusta a los vértices de los polígonos que la conforman, se obtuvo de manera empírica que la separación promedio entre cada dos de estas líneas consecutivas es $N/2$, siendo N la

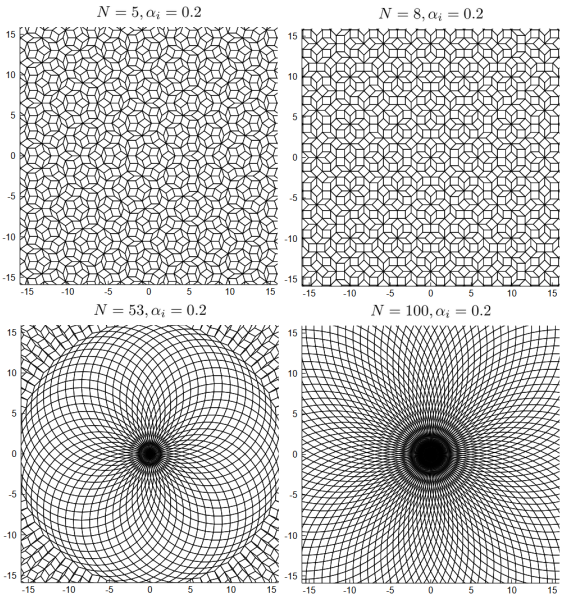

Figura 3.2: Ejemplos de arreglos cuasiperiódicos obtenidos al implementar las expresiones 3.6 - 3.9 considerando todas las parejas válidas de vectores $(\mathbf{e}_j, \mathbf{e}_k)$ y, para cada pareja de vectores, todas las parejas de números enteros (n_j, n_k), donde $n_j, n_k \in [c, d]$ con $c, d \in \mathbb{Z}$ tales que $c < 0 < d$. El número entero N corresponde a la simetría rotacional del arreglo. En todos los casos el parámetro $\alpha_i = 0.2$ empleado fue el mismo para todos los vectores estrella.

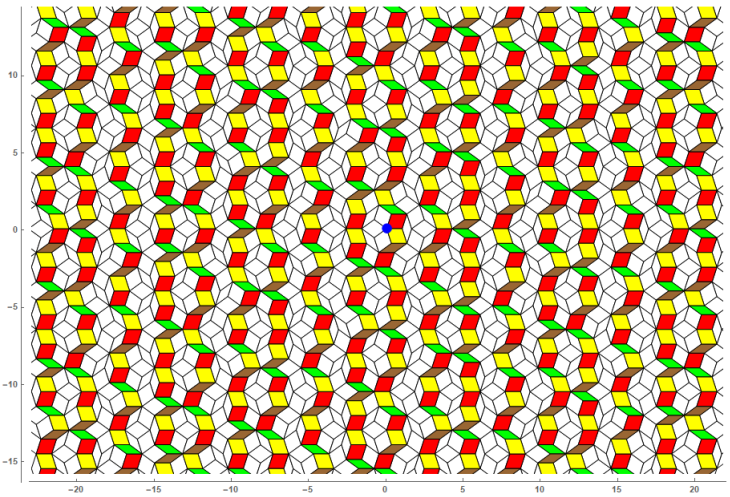

Figura 3.3: Arreglo cuasiperiódico tipo Penrose centrado en el origen. La combinación de vectores estrella $(\mathbf{e}_j, \mathbf{e}_k)$ que dan lugar a los polígonos coloreados es la siguiente: rojo - $(\mathbf{e}_1, \mathbf{e}_2)$, verde - $(\mathbf{e}_1, \mathbf{e}_3)$, café - $(\mathbf{e}_1, \mathbf{e}_4)$ y amarillo - $(\mathbf{e}_1, \mathbf{e}_5)$ siendo el vector $\mathbf{e}_1 = (1, 0)$. Las "franjas" aparecen en el orden del número entero n_1, esto es, la "franja" más cercana por la derecha al origen (denotado por un punto azul) se formó al considerar $n_1 = 0$, la siguiente "franja" a la derecha se formó al considerar $n_1 = 1$ y así sucesivamente (de forma análoga para las "franjas" a la izquierda).

simetría rotacional del arreglo cuasiperiódico. De esta manera, para cada vector estrella $\mathbf{e}_i$, podemos dividir al plano en franjas de ancho $N/2$ ortogonales a dicho vector.

Con base en lo anterior podemos aproximar el número entero n_i asociado al vector estrella $\mathbf{e}_i$ para generar los polígonos del arreglo cuasiperiódico que se encuentren en una vecindad alrededor de un punto $\mathbf{P}$ arbitrario de la siguiente forma: Dado que todo el plano ha sido dividido en franjas de ancho $N/2$, el punto $\mathbf{P}$ está contenido en alguna de estas franjas, supongamos que la franja en la que se encuentra el punto es aquella cuyos lados corresponden a las rectas que mejor aproximan a las "franjas ortogonales" al vector estrella $\mathbf{e}_i$ con números enteros k y $k+1$, de esta manera sabemos que el número entero n_i asociado al vector estrella $\mathbf{e}_i$ que genere los polígonos cercanos al punto $\mathbf{P}$ debe ser alguno de estos dos números.

Para aproximar el valor del número entero asociado al lado más cercano a $\mathbf{P}$ de la franja que lo contiene lo que hacemos es proyectar dicho punto con el vector estrella $\mathbf{e}_i$ y, dado que nuestro vector estrella es unitario por la forma en que se generó, dividir el resultado de dicha proyección por el ancho de las franjas (esto para saber cuántas veces cabe una franja en la proyección del punto $\mathbf{P}$ sobre la dirección en la cual las franjas cubren al plano) y aproximar el resultado al número entero más cercano, es decir

$$n_i = \left\lfloor \frac{2\mathbf{P} \cdot \mathbf{e}_i}{N} \right\rceil, \tag{3.10}$$

donde $\lfloor \cdot \rceil$ denota la función que regresa el entero más cercano.

Notemos que debido a que estamos aproximando los enteros n_i a partir de redondear un número real a su entero más cercano, así como el hecho de que en la expresión 3.10 estamos considerando que la recta que mejor describe a la "franja ortogonal" asociada al entero $n_i = 0$ pasa por el origen (lo cual en general es falso, el qué tanto dista del origen depende del parámetro α_i), es necesario que consideremos un margen de error para cada uno de los números enteros obtenidos por este método.

De esta manera, al momento de generar el polígono asociado a la pareja de vectores estrella $(\mathbf{e}_i, \mathbf{e}_j)$, hay que considerar las parejas de números enteros (m_i, m_k) con $m_l \in [n_l - \beta_l, n_l + \beta_l]$ donde $\beta_l \in \mathbb{Z}^+ \cup \{0\}$ está asociado al margen de error considerado para el número entero n_l. En general, estos parámetros β_l nos permiten determinar qué tan grande será la vecindad de la retícula cuasiperiódica que generemos alrededor del punto arbitrario $\mathbf{P}$: a mayor valor de β_l, mayor el tamaño de la vecindad de la retícula cuasiperiódica generada (véase la figura 3.4).

Una mejor aproximación al cálculo de los números enteros n_i mejoraría la calidad de las vecindades del arreglo cuasiperiódico obtenidas, reduciendo en general el tiempo de cómputo requerido para trabajar con dichas vecindades en sus diferentes aplicacio-

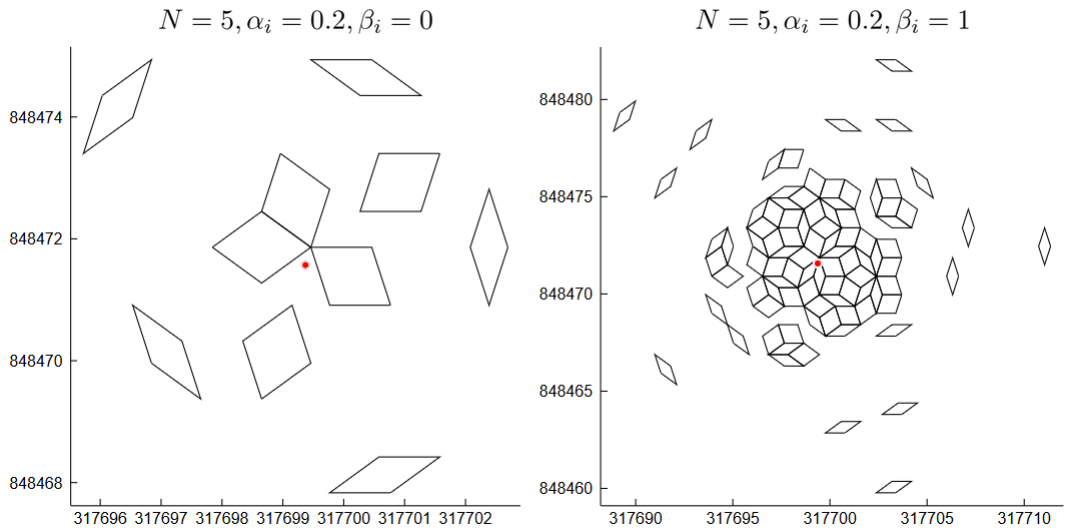

Figura 3.4: Vecindad de un arreglo cuasiperiódico tipo Penrose (simetría orientacional $N = 5$) generada con constantes $\alpha_i = 0.2$ para todo valor de i. La vecindad se obtuvo alrededor del punto rojo con coordenadas $(317699.3711619294, 848471.5693893201)$. En la imagen de la izquierda tenemos los polígonos generados al considerar $\beta_i = 0$ para todo valor de i. En la imagen de la derecha tenemos los generados considerando un margen de error $\beta_i = 1$ para todo valor de i.

nes (véase el capítulo 5 y 6 para conocer más acerca de algunas posibles aplicaciones e implementaciones del presente método). Debido a que el parámetro α_i está relacionado con la distancia respecto al origen de la "franja ortogonal" asociada al entero $n_i = 0$, una mejora a la expresión 3.10 debería incluir dicho parámetro.

De manera empírica hemos encontrado que la expresión

$$n_i = \left\lfloor \frac{2\left(\mathbf{P} - \alpha_i \mathbf{e}_i\right) \cdot \mathbf{e}_i}{N} \right\rfloor, \tag{3.11}$$

reduce en general el error cometido en la aproximación de estos números n_i con lo cual la calidad de la vecindad del arreglo cuasiperiódico mejora (véase la comparativa contenida en la figura 3.5), sin embargo cabe mencionar que esta es una expresión empírica y que en ningún momento se pretende sugerir que el desplazamiento α_i introducido en la expresión 3.1 se traduce directamente como un desplazamiento en el espacio donde vive el arreglo cuasiperiódico.

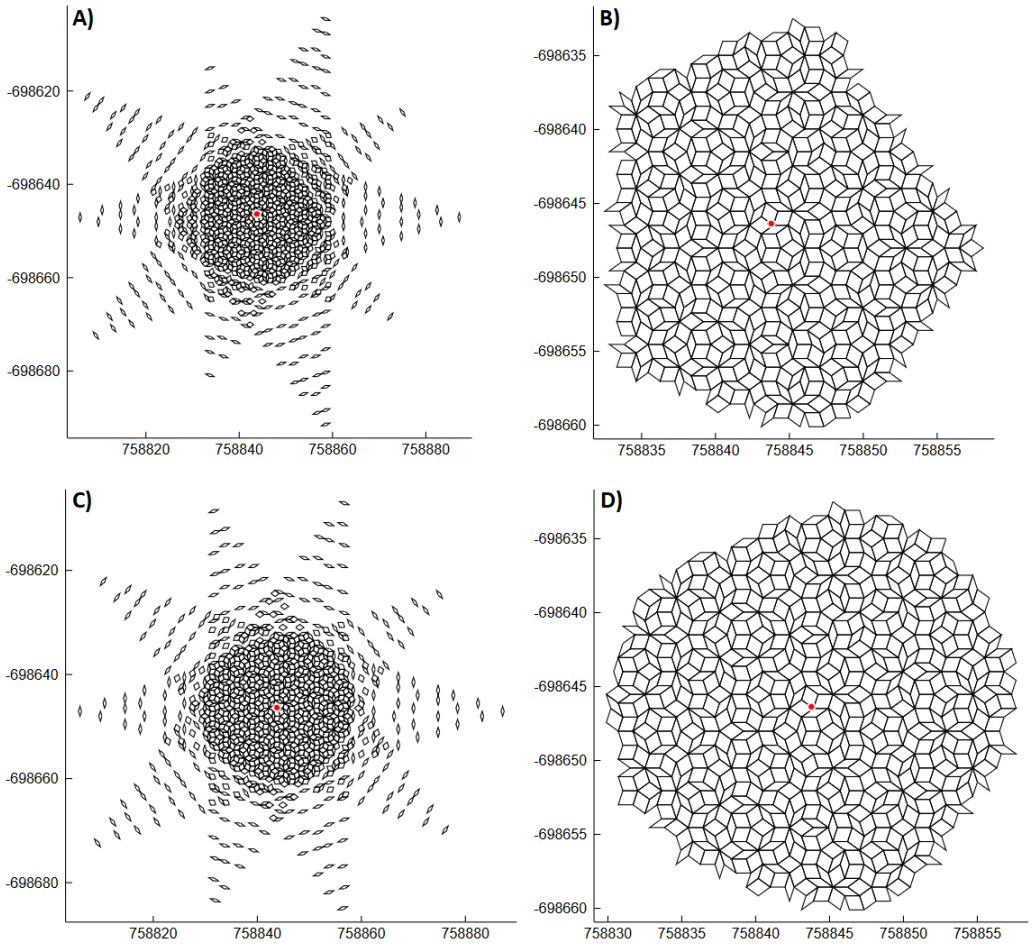

Figura 3.5: Conjunto de polígonos de un arreglo cuasiperiódico con simetría pentagonal, $N = 5$, generado alrededor del punto rojo $\mathbf{P} = (758843.7671146238, -698646.3607070035)$ con las constantes $\alpha_i = 0.567813548$ y $\beta_i = 5\ \forall i \in \{1, 2, \ldots, 5\}$. En la imagen **A** el conjunto de números enteros n_i para generar los polígonos del arreglo cuasiperiódico se aproximó con la expresión 3.10, mientras que en la imagen **C** dicho conjunto se obtuvo con la expresión 3.11. Las imágenes **B** y **D** corresponden al clúster central de polígonos de los conjuntos mostrados en **A** y **C** respectivamente. Nótese que el clúster en **D** cubre más área que el clúster en **B**.

Capítulo 4

Teselados de Voronoi

Planteémonos la siguiente situación hipotética: El gobierno de la ciudad en la cual usted vive ha comenzado un proceso de expansión y mejora a los servicios de seguridad con el objetivo de reducir los índices de criminalidad, para ello lo han contratado a usted como consultor y le han proporcionado los planos de la zona urbana así como la localización de los actuales servicios disponibles, entre los cuales se encuentran varios cuarteles policíacos con unidades terrestres para atender las llamadas de emergencia. Con base en encuestas realizadas a la población para conocer la opinión pública respecto a las ventanas de oportunidad que existen en los actuales servicios de seguridad, usted se propone el mejorar los tiempos de respuesta de la policía desde el momento en que se recibe la denuncia vía telefónica hasta que una unidad se presenta en la escena del crimen, para ello contempla la posibilidad de construir un nuevo cuartel en algún punto de la ciudad, con lo cual le surge la siguiente pregunta: ¿Cuál es el mejor sitio para construir un nuevo cuartel si queremos mejorar los tiempos de respuesta?

De manera intuitiva podemos notar que dicho sitio no debe estar demasiado cerca de alguno de los cuarteles ya existentes, pues en ese caso las zonas a la cuales tengan acceso más rápido las unidades procedentes del nuevo cuartel serán prácticamente las mismas que aquellas correspondientes a las unidades procedentes del cuartel cercano. Suponiendo que existe una distribución homogénea en la incidencia delictiva, un primer paso para conocer el mejor sitio donde colocar el nuevo cuartel es determinar cuál es el área de respuesta asociada a cada uno de los cuarteles ya existente, de esta manera el sitio que buscamos se encontrará dentro del área que resulte más grande. Este procedimiento de determinar el área de respuesta de los cuarteles o, en un caso más general, el área (volumen) de influencia de un conjunto de puntos en el espacio, es lo que da lugar a un teselado de Voronoi.

La manera en la cual se construye dicho teselado depende fuertemente de cómo medimos distancias entre dos puntos dentro del espacio que estamos considerando, por ejemplo para el caso particular desarrollado en párrafos anteriores resulta claro que una métrica Euclidiana no es viable dado que las unidades terrestres están sujetas a despla-

zarse sobre las calles que conforman a la ciudad, por lo que una métrica de Manhattan (también conocida como la métrica del taxista) resulta más adecuada; de igual forma puede ser que no todos los puntos a los que estamos interesados en conocer su área de influencia tengan una misma importancia o peso, por ejemplo en el caso particular que estamos considerando podría ocurrir que algunos de los cuarteles posean unidades terrestres más rápidas con lo cual su área de influencia no sólo estará determinada por su localización espacial, sino también por estas características adicionales.

En general existe una amplia variedad de teselados de Voronoi que se pueden construir al dotar al espacio y/o a los puntos con diferentes características, sin embargo para los fines del presente trabajo nos restringiremos a estudiar los denominados "teselados de Voronoi ordinarios" a los cuales nos referiremos únicamente como teselados de Voronoi dado que no habrá ambigüedad al respecto. El contenido desarrollado a continuación está basado principalmente en los capítulos 1, 2 y 4 del libro "Spatial tessellations: Concepts and Applications of Voronoi Diagrams" [39], cualquier referencia complementaria será mencionada de manera explícita.

4.1. Conceptos preliminares

Hasta este momento se ha mencionado de forma intuitiva la idea detrás de un teselado de Voronoi, sin embargo para poder desarrollar de manera formal este concepto es necesario definir previamente una serie de términos y desarrollar algunos resultados matemáticos que nos serán necesarios más adelante. Salvo que se indique lo contrario, en la presente sección consideraremos únicamente espacios Euclídeos, esto es un espacio vectorial completo $\mathbb{R}^m$ (con $m \in \mathbb{Z}^+$ la dimensión del espacio) dotado con la norma Euclidiana $|\cdot|$.

- **Definición 4.1.1:** Sea S un subconjunto cerrado de $\mathbb{R}^m$, S_i un subconjunto cerrado de S y $\varsigma = \{S_1, \ldots, S_n\}$. Si los elementos del conjunto ς satisfacen

$$(S_i \setminus \partial S_i) \cap (S_j \setminus \partial S_j) = \emptyset, \quad i \neq j,\ i, j \in \{1, \ldots, n\}, \tag{4.1}$$

$$\bigcup_{i=1}^{n} S_i = S, \tag{4.2}$$

  entonces decimos que el conjunto ς es una ***teselación*** de S.

- **Definición 4.1.2:** Sea $A \subseteq \mathbb{R}^m$ un conjunto de puntos no vacío. Si para cualesquiera dos puntos $\mathbf{x}_1, \mathbf{x}_2 \in A$ el segmento que los une es un subconjunto de A, es decir se cumple que

$$\lambda \mathbf{x}_1 + (1 - \lambda)\, \mathbf{x}_2 \in A, \quad \forall \lambda \text{ tal que } 0 \leq \lambda \leq 1, \tag{4.3}$$

entonces el conjunto A es un ***conjunto convexo***; en caso contrario el conjunto A es un ***conjunto no convexo***.

- **Teorema 4.1.1:** La intersección de dos conjuntos convexos es un conjunto convexo.

  Demostración: Sean $A, B \subseteq \mathbb{R}^m$ dos conjuntos convexos. Si $A \cap B = \emptyset$ o si $A \cap B = \{\mathbf{x}_1\}$ para algún punto $\mathbf{x}_1 \in \mathbb{R}^m$, se satisface por vacuidad que el conjunto $A \cap B$ es convexo. Supongamos entonces que $A \cap B$ es un conjunto no vacío con al menos dos elementos diferentes.

  Sean $\mathbf{x}_1, \mathbf{x}_2 \in A \cap B$ dos puntos arbitrarios distintos, como por hipótesis estos viven en la intersección de A con B, en particular viven en el conjunto A y en el conjunto B, es decir $\mathbf{x}_1, \mathbf{x}_2 \in A$ y $\mathbf{x}_1, \mathbf{x}_2 \in B$. Dado que por hipótesis los conjuntos A y B son convexos, se sigue por la relación 4.3 que $\lambda\mathbf{x}_1 + (1-\lambda)\,\mathbf{x}_2 \in A$ y $\lambda\mathbf{x}_1 + (1-\lambda)\,\mathbf{x}_2 \in B$ para toda $\lambda \in [0,1]$, por lo tanto se satisface que $\lambda\mathbf{x}_1 + (1-\lambda)\,\mathbf{x}_2 \in A \cap B$ para toda $\lambda \in [0,1]$ con lo que, por definición, se demuestra que el conjunto $A \cap B$ es convexo.

- **Definición 4.1.3:** Sea $\mathbb{R}^m$ un espacio Euclídeo. Denominamos al conjunto de puntos

$$P = \{\mathbf{x} \mid \mathbf{x} \cdot \mathbf{n} = b;\ \mathbf{n}, \mathbf{x} \in \mathbb{R}^m, b \in \mathbb{R}, \mathbf{n} \neq \vec{0}\}, \tag{4.4}$$

  un ***hiperplano*** de dimensión $m-1$.

- **Definición 4.1.4:** Sea $\mathbb{R}^m$ un espacio Euclídeo. Definimos al conjunto de puntos

$$H = \{\mathbf{x} \mid \mathbf{x} \cdot \mathbf{n} < b;\ \mathbf{n}, \mathbf{x} \in \mathbb{R}^m, b \in \mathbb{R}, \mathbf{n} \neq \vec{0}\}, \tag{4.5}$$

  el ***semiespacio*** generado por el hiperplano P.

- **Definición 4.1.5:** Dado un espacio Euclídeo $\mathbb{R}^m$ definimos a los ***poliedros*** como todo conjunto de puntos no vacío, conexo y acotado construido a partir de un número finito de intersecciones y uniones de una cantidad finita de semiespacios. Al caso particular en el que el poliedros es un conjunto convexo se le denomina ***politopo***.

4.2. Propiedades básicas del teselado de Voronoi

En la presente sección estudiaremos, a partir de las definiciones y teoremas presentados en la sección previa, las principales características que poseen los teselados de Voronoi en un espacio Euclídeo. Comencemos definiendo formalmente al teselado de Voronoi generado por una cantidad finita de puntos en $\mathbb{R}^m$.

- **Definición 4.2.1 (Teselado de Voronoi):** Sea $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{R}^m$ donde $2 < n < \infty$ y $\mathbf{p}_i \neq \mathbf{p}_j$ si $i \neq j,\ i, j \in \{1, \ldots, n\}$. Llamamos a la región definida por
$$V(\mathbf{p}_i) = \{\mathbf{x} \mid |\mathbf{x} - \mathbf{p}_i| \leq |\mathbf{x} - \mathbf{p}_j|,\ \forall j \neq i, j \in \{1, \ldots, n\}\}, \tag{4.6}$$
la ***celda de Voronoi*** asociada al punto $\mathbf{p}_i$ (o el ***polígono de Voronoi*** asociado al punto $\mathbf{p}_i$ en el caso bidimensional) y al conjunto definido como
$$\mathbb{V} = \{V(\mathbf{p}_1), \ldots, V(\mathbf{p}_n)\}, \tag{4.7}$$
el ***diagrama (teselado) de Voronoi*** generado por el conjunto P.

A cada uno de los puntos $\mathbf{p}_i \in P$ se le denomina el ***punto generador*** de la i-ésima celda de Voronoi con lo que al conjunto P se le conoce como el ***conjunto generador***.

Notemos que a partir de la expresión 4.6 cada celda de Voronoi queda definida como un conjunto cerrado al incluir su frontera $\partial V(\mathbf{p}_i)$, para el caso particular donde el espacio es $\mathbb{R}^2$ dicha frontera puede estar conformada por segmentos y/o semirectas que se denominan **bordes de Voronoi** cuyas fronteras a su vez se denominan **vértices de Voronoi**, en dimensiones más altas las fronteras de cada celda de Voronoi consisten en otros subconjuntos del espacio $\mathbb{R}^m$ como semiplanos, regiones acotadas de un hiperplano, etcétera.

De manera formal, a la frontera de una celda de Voronoi m-dimensional se le denomina ***cara de Voronoi*** $(m-1)$***-dimensional***, a la frontera de este último conjunto se le denomina ***cara de Voronoi*** $(m-2)$***-dimensional***, y así sucesivamente, hasta llegar a la frontera de una cara de Voronoi 2-dimensional la cual llamamos bordes de Voronoi y cuya frontera denominamos vértices de Voronoi. En general, si dos celdas de Voronoi satisfacen que su intersección es no vacía, esto es $V(\mathbf{p}_i) \cap V(\mathbf{p}_j) \neq \emptyset$, entonces el conjunto de puntos resultado de dicha intersección define a la cara de Voronoi $(m-1)$-dimensional compartida por ambas celdas.

- **Definición 4.2.2:** Sea $e(\mathbf{p}_i, \mathbf{p}_j) = V(\mathbf{p}_i) \cap V(\mathbf{p}_j)$ para cualesquiera dos puntos $\mathbf{p}_i, \mathbf{p}_j \in P$ con $i \neq j$. Si el conjunto $e(\mathbf{p}_i, \mathbf{p}_j)$ es no vacío y está conformado por más de un punto, decimos que las celdas de Voronoi $V(\mathbf{p}_i)$ y $V(\mathbf{p}_j)$ son ***vecinas*** o ***adjacentes***.

Una característica importante de las celdas de Voronoi es el hecho de que son conjuntos convexos, para poder demostrar esto de manera sencilla reescribamos la definición de las celdas de Voronoi en términos de semiplanos.

Sean $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^m$ dos puntos generadores de un teselado de Voronoi y sea $m(\mathbf{p}_i, \mathbf{p}_j)$ el conjunto de puntos dado por
$$m(\mathbf{p}_i, \mathbf{p}_j) = \{x \mid |\mathbf{x} - \mathbf{p}_i| = |\mathbf{x} - \mathbf{p}_j|,\ j \neq i\}, \tag{4.8}$$

a este conjunto se le conoce como la ***mediatriz*** definida por los puntos $\mathbf{p}_i$ y $\mathbf{p}_j$. Mostremos explícitamente que dicho conjunto es un hiperplano, para ello desarrollemos la condición que satisfacen los puntos que lo conforman:

$$|\mathbf{x} - \mathbf{p}_i| = |\mathbf{x} - \mathbf{p}_j|,$$

$$\Rightarrow \sqrt{(\mathbf{x} - \mathbf{p}_i) \cdot (\mathbf{x} - \mathbf{p}_i)} = \sqrt{(\mathbf{x} - \mathbf{p}_j) \cdot (\mathbf{x} - \mathbf{p}_j)},$$

$$\Rightarrow \mathbf{x} \cdot \mathbf{x} - 2\left(\mathbf{x} \cdot \mathbf{p}_i\right) + \mathbf{p}_i \cdot \mathbf{p}_i = \mathbf{x} \cdot \mathbf{x} - 2\left(\mathbf{x} \cdot \mathbf{p}_j\right) + \mathbf{p}_j \cdot \mathbf{p}_j,$$

$$\Rightarrow \mathbf{x} \cdot \mathbf{p}_j - \frac{1}{2}\left(\mathbf{p}_j \cdot \mathbf{p}_j\right) = \mathbf{x} \cdot \mathbf{p}_i - \frac{1}{2}\left(\mathbf{p}_i \cdot \mathbf{p}_i\right),$$

$$\Rightarrow \mathbf{x} \cdot \mathbf{p}_j - \mathbf{x} \cdot \mathbf{p}_i = \frac{1}{2}\left(\mathbf{p}_j \cdot \mathbf{p}_j - \mathbf{p}_i \cdot \mathbf{p}_i\right),$$

$$\Rightarrow \mathbf{x} \cdot \left(\mathbf{p}_j - \mathbf{p}_i\right) = \frac{1}{2}\left(\mathbf{p}_i \cdot \mathbf{p}_j - \mathbf{p}_i \cdot \mathbf{p}_i + \mathbf{p}_j \cdot \mathbf{p}_j - \mathbf{p}_j \cdot \mathbf{p}_i\right),$$

$$\Rightarrow \mathbf{x} \cdot \left(\mathbf{p}_j - \mathbf{p}_i\right) = \frac{1}{2}\left(\mathbf{p}_i + \mathbf{p}_j\right) \cdot \left(\mathbf{p}_j - \mathbf{p}_i\right),$$

definamos a los vectores $\mathbf{u} = \mathbf{p}_j - \mathbf{p}_i$ y $\mathbf{v} = \frac{1}{2}\left(\mathbf{p}_i + \mathbf{p}_j\right)$

$$\Rightarrow \mathbf{x} \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{u},$$

$$\Rightarrow \mathbf{x} \cdot \mathbf{u} = k,$$

donde $k = \mathbf{v} \cdot \mathbf{u}$ es una constante real. De esta manera, podemos reescribir la expresión 4.8 como

$$m(\mathbf{p}_i, \mathbf{p}_j) = \{x \mid \mathbf{x} \cdot \mathbf{u} = k\}, \tag{4.9}$$

la cual es, según la definición 4.1.3, la expresión para un hiperplano $m-1$ dimensional. Nótese que la condición $\mathbf{u} \neq \vec{0}$ se satisface dado que los puntos generadores $\mathbf{p}_i$ y $\mathbf{p}_j$ son diferentes, con lo cual su resta no puede ser el vector cero.

Sea $H(\mathbf{p}_i, \mathbf{p}_j)$ el semiespacio generado por la mediatriz $m(\mathbf{p}_i, \mathbf{p}_j)$, esto es

$$H\left(\mathbf{p}_i, \mathbf{p}_j\right) = \{\mathbf{x} \mid |\mathbf{x} - \mathbf{p}_i| \leq |\mathbf{x} - \mathbf{p}_j|,\ j \neq i\}. \tag{4.10}$$

El mismo desarrollo realizado para mostrar que la mediatriz es un hiperplano permite mostrar que el conjunto $H\left(\mathbf{p}_i, \mathbf{p}_j\right)$ es efectivamente un semiespacio.

Denominamos a la región $H\left(\mathbf{p}_i, \mathbf{p}_j\right)$ la región de dominio de $\mathbf{p}_i$ sobre $\mathbf{p}_j$ y esta corresponde a todos los puntos del espacio que se encuentran más cerca (o en el peor de los casos a una misma distancia) del punto $\mathbf{p}_i$ que del punto $\mathbf{p}_j$. Notemos como

el conjunto $H\left(\mathbf{p}_i, \mathbf{p}_j\right)$ corresponde a una de las condiciones que satisfacen los puntos que conforman una celda de Voronoi según la expresión 4.6, de tal forma que podemos expresar a la celda de Voronoi generada por el punto $\mathbf{p}_i$ de la siguiente manera

$$V\left(\mathbf{p}_i\right) = \bigcap_{j \neq i} H\left(\mathbf{p}_i, \mathbf{p}_j\right). \tag{4.11}$$

De esta forma una celda de Voronoi está definida como una intersección finita de un conjunto finito de semiespacios, esto es un poliedro y, dado que los semiespacios son conjuntos convexos, por el teorema 4.1.1 tenemos que su intersección es convexa, con lo cual la celda de Voronoi es un politopo, esto es un poliedro convexo.

4.3. Algoritmos computacionales

Para los fines del presente trabajo estamos no sólo interesados en definir formalmente a los teselados de Voronoi, sino en poder generarlos de manera eficiente a través de algún algoritmo computacional. En la presente sección revisaremos algunos de los aspectos fundamentales acerca de la complejidad computacional y estudiaremos unos cuántos métodos computacionales con los cuales generar teselados de Voronoi en el plano $\mathbb{R}^2$.

Durante el resto del capítulo consideraremos como ciertas las siguientes hipótesis:

- **Hipótesis 1:** Todo cálculo numérico realizado por una computadora se realiza con total precisión aritmética.
- **Hipótesis 2:** Los puntos generadores son tales que ningún círculo pasa por cuatro de ellos.

La hipótesis 1 es necesaria para poder estudiar de manera teórica a los algoritmos computacionales, básicamente lo que nos indica es que todo error numérico será relegado a estudiarse por separado como parte de la estructura del procesador que correrá dichos algoritmos y no como parte inherente al algoritmo en sí.

La hipótesis 2 es introducida para evitar que los teselados de Voronoi sean degenerados, esto es que haya más de tres celdas de Voronoi que compartan un mismo vértice.

Estas dos hipótesis serán comunes al desarrollo de los algoritmos que se estudiarán más adelante dentro de este capítulo, con la posibilidad de añadir alguna hipótesis adicional necesaria para cada algoritmo en particular.

4.3.1. Complejidad computacional

Una de las cuestiones más importantes al momento de desarrollar un algoritmo computacional es que este sea "eficiente", es decir que el tiempo de cómputo necesario para que el algoritmo finalice sea el menor posible. Una manera estándar para determinar qué tan eficiente es un algoritmo consiste en observar el comportamiento asintótico del tiempo que este requiere en ser completado como función del tamaño de los datos que conforman a los parámetros de entrada del algoritmo.

Sea $T(n)$ el tiempo requerido por un algoritmo en ser completado cuando el tamaño de los datos de entrada proporcionados es n (por ejemplo, un arreglo con n números). Debido a que, en principio, existe una amplia variedad de datos de tamaño n que son válidos como parámetros de entrada del algoritmo a considerar y cada uno de ellos tardará, de manera general, un tiempo diferente en ser procesado, existen dos principales formas de estimar la función $T(n)$: La primera de ellas, el tiempo del peor escenario, consiste en definir a $T(n)$ como el máximo tiempo que requiere el algoritmo considerando todos los posibles datos de entrada de tamaño n, mientras que en la segunda forma, el tiempo promedio, se define a $T(n)$ como el promedio de los tiempos requeridos por el algoritmo considerando todos los posibles datos de entrada de tamaño n.

Sea $f(x)$ una función definida sobre los números reales positivos tal que $f(x) > 0$ para todo valor de x. Si existen constantes $C, N \in \mathbb{R}$ tal que

$$\frac{T\,(n)}{f\,(n)} < C,\ \forall n \geq N, \tag{4.12}$$

decimos que $T\,(n)$ es de orden $f\,(n)$ y lo denotamos como $T\,(n) = O\,(f\,(n))$, en cuyo caso decimos que la complejidad temporal del algoritmo es $O\,(f\,(n))$.

La desigualdad 4.12 nos indica que, a partir de un cierto tamaño de los datos proporcionados, el tiempo requerido por el algoritmo para ser llevado a cabo no incrementará más rápido que la manera en que la función $f\,(n)$ crece conforme aumenta la variable n.

De esta forma, un algoritmo que satisface que $T\,(n) = O(1)$ es "idealmente eficiente" al no depender su tiempo de cómputo del tamaño de los datos proporcionados. A los algoritmos con $T\,(n) = O(n)$ se les denomina **algoritmos con tiempo lineal** mientras que a los algoritmos con $T\,(n) = O(n^q)$ se les denomina **algoritmos con tiempo polinomial**.

4.3.2. Divide y vencerás

Una de las técnicas más importantes y más empleadas para diseñar algoritmos eficientes es la que lleva por nombre "divide y vencerás" [40]. Partamos del escenario en

el cual tenemos un problema a resolver, al cual denominaremos "problema central"; la técnica "divide y vencerás" consiste básicamente en ir dividiendo el problema central en problemas más pequeños (generalmente más sencillos de resolver) de tal forma que, a partir de las soluciones obtenidas para estos problemas más pequeños se pueda construir la solución al problema inicial.

Introduzcamos esta técnica aplicándola al problema de multiplicar números enteros binarios. Sean X y Y dos números enteros en base dos, cada uno con n dígitos; el algoritmo usual que se enseña en la educación básica para multiplicar dos números enteros (el cual es igualmente válido para el producto de números enteros binarios) involucra el cálculo de n productos parciales, cada uno de ellos de tamaño n, con lo cual dicho algoritmo tiene un tiempo de orden $O(n^2)$ si consideramos a las sumas y productos de un sólo dígito como operaciones de tiempo constante.

Podemos obtener un nuevo algoritmo para calcular el producto de X y Y implementando la técnica "divide y vencerás", para ello expresemos cada uno de los números enteros en términos de dos números enteros de $n/2$ dígitos (por simplicidad supongamos que la cantidad de dígitos n es una potencia de 2), esto es

$$X = A2^{n/2} + B,\ Y = C2^{n/2} + D,$$

donde A, B, C y D son números de $n/2$ dígitos. De esta manera el producto de X con Y está dado por

$$XY = (AC)2^n + (AD + BC)\,2^{n/2} + BD. \tag{4.13}$$

Si evaluamos el producto XY a través de este procedimiento, tendremos que calcular ahora cuatro multiplicaciones de números enteros de $n/2$ dígitos (los productos AC, AD, BC y BD), tres sumas de números enteros con a los más $2n$ dígitos y dos productos por las constantes 2^n y $2^{n/2}$. Considerando que los productos por constantes y las sumas son operaciones de tiempo lineal, esto es $O(n)$, tenemos que el tiempo requerido por este algoritmo está dado por

$$T(n) = 4T\left(\frac{n}{2}\right) + cn, \tag{4.14}$$

para alguna constante positiva c. Ahora, para calcular el producto de los números enteros con $n/2$ dígitos procedemos de la misma manera, esto es expresar cada uno de dichos números en términos de dos números de $n/4$ dígitos y proceder a realizar los respectivos productos de estos nuevos números enteros. El proceso es iterativo hasta que lleguemos a productos de números enteros con un sólo dígito, los cuales son operaciones de tiempo constante.

Expandiendo la expresión 4.14 tenemos

$$T(n) = 4\left[4T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] + cn = 4^2T\left(\frac{n}{2^2}\right) + (2+1)cn,$$

$$\Rightarrow T(n) = 4^2\left[4T\left(\frac{n}{2^3}\right) + c\frac{n}{2^2}\right] + (2+1)cn = 4^3T\left(\frac{n}{2^3}\right) + (2^2+2+1)cn,$$

por inducción tenemos que en la última iteración i-ésima la expresión para $T(n)$ es

$$T(n) = 4^iT\left(\frac{n}{2^i}\right) + \sum_{j=0}^{i-1} 2^j cn, \tag{4.15}$$

donde, si es la última iteración, el tiempo requerido para calcular cada uno de los productos es constante, esto es

$$T\left(\frac{n}{2^i}\right) = T(1), \Rightarrow \frac{n}{2^i} = 1, \Rightarrow n = 2^i,$$

por otro lado tenemos que

$$\sum_{j=0}^{i-1} 2^j = 2^i - 1,$$

sustituyendo estos resultados en la expresión 4.15 tenemos

$$T(n) = n^2T(1) + (n-1)cn = kn^2 + cn(n-1),$$

para algunas constantes positivas c y k de donde obtenemos que

$$T(n) = O(n^2). \tag{4.16}$$

Este resultado es importante porque nos muestra que el uso de la técnica "divide y vencerás" no necesariamente representa una mejora en la eficiencia del algoritmo, sino que se requiere además tener una buena estrategia y/o el ingenio de proponer un nuevo enfoque al problema. Relacionado a este punto, Karatsuba y Ofman propusieron en 1963 un nuevo algoritmo con el cual, implementando la técnica "divide y vencerás", la eficiencia sí mejora [41].

A grandes rasgos la propuesta de Karatsuba y Ofman consiste en reescribir la expresión 4.13 como

$$XY = (AC)2^n + \left[(A-B)(D-C) + AC + BD\right]2^{n/2} + BD, \tag{4.17}$$

donde, pese a que en principio parece ser más complicada, el punto clave radica en el hecho de que ahora sólo se necesita calcular tres multiplicaciones de números enteros de $n/2$ dígitos (los productos AC, BD y $(A-B)(D-C)$) contrario a las cuatro multiplicaciones que teníamos previamente. Esto viene con el coste de que ahora debemos considerar seis operaciones de suma o sustracción y dos productos por las constantes 2^n y $2^{n/2}$, sin embargo, dado que las sumas o sustracciones, así como los productos por constantes son operaciones de tiempo lineal, el tiempo requerido por este algoritmo está expresado por

$$T(n) = 3T\left(\frac{n}{2}\right) + cn, \tag{4.18}$$

para alguna constante positiva c. Expandiendo esta expresión iterativa tenemos

$$T(n) = 3\left[3T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] + cn = 3^2T\left(\frac{n}{2^2}\right) + \left(1+\frac{3}{2}\right)cn,$$

$$\Rightarrow T(n) = 3^2\left[3T\left(\frac{n}{2^3}\right) + c\frac{n}{2^2}\right] + \left(1+\frac{3}{2}\right)cn = 3^3T\left(\frac{n}{2^3}\right) + \left(1+\frac{3}{2}+\frac{3^2}{2^2}\right)cn,$$

por inducción tenemos que en la última iteración i-ésima la expresión para $T(n)$ es

$$T(n) = 3^iT\left(\frac{n}{2^i}\right) + \sum_{j=0}^{i-1}\left(\frac{3}{2}\right)^j cn, \tag{4.19}$$

donde, si es la última iteración, el tiempo requerido para calcular cada uno de los productos es constante, esto es

$$T\left(\frac{n}{2^i}\right) = T(1),\ \Rightarrow \frac{n}{2^i} = 1,\ \Rightarrow n = 2^i,$$

ahora notemos que

$$3^i = \left(2^{\log(3)}\right)^i = 2^{i\log(3)} = (2^i)^{\log(3)} = n^{\log(3)} \approx n^{1.585},$$

y

$$\sum_{j=0}^{i-1}\left(\frac{3}{2}\right)^j = \frac{\left(\frac{3}{2}\right)^i - 1}{\frac{3}{2}-1} = 2\frac{3^i}{2^i} - 2 = 2n^{\log(3)-1} - 2,$$

de tal forma que la expresión 4.19 se vuelve

$$T(n) = n^{\log(3)}T(1) + 2cn^{\log(3)} - 2cn = kn^{\log(3)} + bn^{\log(3)} - bn,$$

para algunas constantes positivas b y k. De la expresión anterior tenemos que

$$T(n) = O(n^{\log(3)}). \tag{4.20}$$

4.3.3. Método ingenuo

A finales de la sección 4.2 mostramos como la celda de Voronoi relacionada con el punto generador $\mathbf{p}_i$ puede ser construida a través de la intersección de un número finito de semiespacios (semiplanos para el caso dos dimensional), cada uno de ellos asociado a la mediatriz definida por una pareja de puntos $\mathbf{p}_i$ y $\mathbf{p}_j$, con $j \neq i$, del conjunto generador P. Esta caracterización de las celdas de Voronoi da lugar al método más directo (al cual nos referiremos como el método ingenuo) para generar un teselado de Voronoi en dos dimensiones.

Sea $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ un conjunto generador, el algoritmo que conforma al método ingenuo es el siguiente:

Algoritmo método ingenuo

1. Se fija el índice i en alguno de los valores del conjunto $\{1, 2, \ldots, n\}$ y se generan los $n-1$ semiplanos $H(\mathbf{p}_i, \mathbf{p}_j)$ con $j \neq i$.

2. Se selecciona uno de los semiplanos $H(\mathbf{p}_i, \mathbf{p}_j)$ generados en el paso anterior y se intersecta dicho conjunto con otro de los semiplanos $H(\mathbf{p}_i, \mathbf{p}_k)$, donde $k \neq j$.

3. El resultado de la intersección del paso anterior se intersecta con otro de los planos generados en el paso 1. Se itera este proceso hasta agotarse todos los semiplanos. El resultado final corresponde a la celda de Voronoi $V(\mathbf{p}_i)$.

4. Se iteran los pasos 1 - 3 sobre los diferentes valores del índice i, el resultado final corresponde al Teselado de Voronoi generado por el conjunto P.

Construir el semiplano $H(\mathbf{p}_i, \mathbf{p}_j)$ dado los puntos $\mathbf{p}_i$ y $\mathbf{p}_j$ requiere un tiempo constante, de tal forma que para cada punto $\mathbf{p}_i$ el tiempo requerido para construir los $n-1$ semiplanos $H(\mathbf{p}_i, \mathbf{p}_j)$, con $j \neq i$, es $a(n-1)$ para alguna constante positiva a.

Analicemos ahora el proceso con el cual obtenemos la celda de Voronoi como resultado de intersectar todos los semiplanos. Notemos que en una primera instancia al intersectar dos semiplanos arbitrarios $H(\mathbf{p}_i, \mathbf{p}_j)$ y $H(\mathbf{p}_i, \mathbf{p}_k)$ con $k \neq j$ obtenemos un polígono con dos lados (salvo que uno de los semiplanos esté completamente contenido en el otro, lo cual ocurre únicamente si las bisectrices al segmento definido por cada pareja de puntos son paralelas), después intersectamos este polígono resultante con otro de los semiplanos disponibles y así proseguimos. En general, durante el k-ésimo paso de este procedimiento, tendremos que encontrar, en el peor de los casos, la intersección de un polígono de k lados con un semiplano (dependiendo del orden de los semiplanos

que vayamos intersectando puede que en el k-ésimo paso tengamos un polígono con una menor cantidad de lados), proceso que requiere de un tiempo proporcional a k dado que es necesario revisar cuáles de los k lados intersectan la frontera del semiplano. Con base en lo anterior tenemos que el tiempo necesario para intersectar los $n-1$ semiplanos es proporcional, en el peor de los casos, a la suma $1+2+\cdots+(n-2)=(n-2)(n-1)/2$, esto es $b(n-2)(n-1)$ para alguna constante positiva b.

Este proceso es para obtener únicamente una celda de Voronoi, por lo cual, el tiempo requerido para generar las n celdas de Voronoi que conforman al teselado de Voronoi por este método es

$$T(n) = n\left[a\,(n-1) + b\,(n-2)\,(n-1)\right] = O(n^3). \tag{4.21}$$

Este método ingenuo puede ser mejorado en cuanto a su eficiencia si la intersección de los $n-1$ semiplanos se realiza empleando un algoritmo basado en la técnica "divide y vencerás", el tiempo empleado por este algoritmo resulta ser del orden $O(n\log n)$, de tal forma que el tiempo $T(n)$ del método ingenuo se vuelve de orden $O(n^2\log n)$.

Veamos a continuación el algoritmo que permite obtener la intersección de $n-1$ semiplanos empleando la técnica "divide y vencerás". Una exposición más detallada de este algoritmo, así como una serie de teoremas y corolarios demostrados a partir del mismo, puede consultarse en la sección 7.2.4 del libro "Computational Geometry: An Introduction" en su segunda impresión [42].

Sea $\{H_1, H_2, \ldots, H_N\}$ un conjunto con N semiplanos contenidos en $\mathbb{R}^2$. Estamos interesados en determinar la región del espacio resultado de intersectar dichos semiplanos, esto es

$$H_1 \cap H_2 \cap \cdots \cap H_N;$$

dado que la intersección de conjuntos es una operación asociativa, los términos de la expresión anterior pueden ser reagrupados de la siguiente manera

$$\left(H_1 \cap \cdots \cap H_{N/2}\right) \cap \left(H_{N/2+1} \cap \cdots \cap H_N\right), \tag{4.22}$$

donde, si N no es un número par, sustituimos el subíndice $N/2$ por $\lfloor N/2\rfloor$ y el subíndice $N/2+1$ por $\lceil N/2\rceil$.

El resultado de intersectar los semiplanos contenidos en el paréntesis de la izquierda nos dará un polígono convexo (no necesariamente cerrado) con a lo más una cantidad $N/2$ (o $\lfloor N/2\rfloor$ si el número de semiplanos es impar) de lados; de forma análoga ocurrirá para la intersección de los semiplanos contenidos en el paréntesis de la derecha.

Empleando un algoritmo proporcionado por O'Rourke, Chien, Olson y Naddor [43]

se puede mostrar que la intersección de un polígono convexo de L lados con un polígono convexo de M lados puede encontrarse en un tiempo de orden $O(L+M)$. Los detalles de dicho algoritmo, si bien no son complicados para el caso no degenerado (es decir, cuando los polígonos a intersectar no comparten ningún vértice), resultan bastante extensos para ser reproducidos aquí, por lo cual para el lector interesado se recomienda leer la sección 7.2.1 del libro "Computational Geometry: An Introduction" así como el artículo original previamente citado.

A partir de lo expuesto en el párrafo anterior, tenemos que la intersección de los dos polígonos convexos obtenidos tras realizar las intersecciones de cada paréntesis se realiza en un tiempo de orden $O(N)$. Esto sugiere el siguiente algoritmo recursivo para obtener la intersección de N semiplanos:

Intersección de N semiplanos.

1. Si la cantidad de semiplanos a intersectar es dos, se realiza la intersección de manera directa y se regresa al usuario; caso contrario se procede a los siguientes puntos.
2. Se divide al conjunto de semiplanos en dos conjuntos de igual (o aproximadamente igual) cardinalidad.
3. Se obtiene, empleando este mismo algoritmo, la intersección de los semiplanos en cada conjunto generado durante el paso anterior.
4. Se intersectan las regiones resultantes tras la intersección de los semiplanos de cada conjunto generado en el paso 2 y se regresa al usuario dicha región.

Ejemplifiquemos el algoritmo anterior para el caso particular en que queremos intersectar 6 semiplanos, esto es:

$$H_1 \cap H_2 \cap H_3 \cap H_4 \cap H_5 \cap H_6,$$

comencemos expresando este problema inicial como dos subproblemas más sencillos, esto es

$$(H_1 \cap H_2 \cap H_3) \cap (H_4 \cap H_5 \cap H_6)\,,$$

cada uno de estos subproblemas pueden ser expresados, a su vez, como otros dos subproblemas más elementales

$$([H_1 \cap H_2] \cap H_3) \cap ([H_4 \cap H_5] \cap H_6)\,,$$

dado que ya no se puede dividir ninguno de los subproblemas generados en el paso anterior en uno más elemental, se procede a solucionarlos.

Sea $H_{12} = H_1 \cap H_2$ y sea $H_{45} = H_4 \cap H_5$, esto da solución a los dos subproblemas más elementales de tal forma que ahora el problema inicial queda expresado como

$$(H_{12} \cap H_3) \cap (H_{45} \cap H_6),$$

sea $H_{123} = H_{12} \cap H_3$ y $H_{456} = H_{45} \cap H_6$ las respectivas soluciones a los siguientes dos subproblemas, sustituyendo estos conjuntos en la expresión correspondiente tenemos que el problema inicial se ha reducido a obtener la intersección de los dos polígonos convexos

$$H_{123} \cap H_{456}.$$

Con base en lo anterior, si $T(N)$ denota el tiempo empleado para obtener la intersección de N semiplanos por este algoritmo recursivo, tenemos que

$$T(N) = 2T\left(\frac{N}{2}\right) + kN,$$

para alguna constante positiva k. Desarrollemos esta expresión recursiva

$$\Rightarrow T(N) = 2\left[2T\left(\frac{N}{2^2}\right) + k\frac{N}{2}\right] + kN = 2^2T\left(\frac{N}{2^2}\right) + 2kN,$$

$$\Rightarrow T(N) = 2^2\left[2T\left(\frac{N}{2^3}\right) + k\frac{N}{2^2}\right] + 2kN = 2^3T\left(\frac{N}{2^3}\right) + 3kN,$$

por inducción tenemos que, en la última iteración i-ésima la expresión está dada por

$$T(N) = 2^iT\left(\frac{N}{2^i}\right) + ikN, \tag{4.23}$$

si la i-ésima iteración es la última, esto implica que el tiempo requerido para llevar a cabo cada una de esas operaciones es constante, esto es $T\left(\frac{N}{2^i}\right) = T(1)$ de donde tenemos la igualdad

$$\frac{N}{2^i} = 1, \Rightarrow 2^i = N, \Rightarrow i = \log(N),$$

sustituyendo estos resultados en la expresión 4.23 tenemos

$$T(N) = NT(1) + \log(N)kN = cN + kN\log(N),$$

para algunas constantes positivas c y k, con lo que finalmente concluimos que

$$T(N) = O(N\log(N)). \tag{4.24}$$

4.3.4. Método Divide y Vencerás

En la presente sección estudiaremos un método basado en la técnica "divide y vencerás" para construir un teselado de Voronoi. Este método requiere, además de las hipótesis 1 y 2 presentadas al inicio de la sección 4.3, la siguiente hipótesis de trabajo

- **Hipótesis 3**: El conjunto generador $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ es tal que ningún par de puntos generadores se encuentran alineados verticalmente; es decir, $\mathbf{p}_{i_x} \neq \mathbf{p}_{j_x}$ para cualesquiera índices $i, j \in \{1, 2, \ldots, n\}$ con $i \neq j$.

Lo anterior es necesario debido a que, como paso preliminar a la implementación del algoritmo recursivo que define al presente método, se requiere reacomodar a los puntos generadores $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ en orden creciente respecto a su coordenada x, con lo cual la hipótesis 3 nos asegura que dicho orden es único. Este reacomodo del conjunto generador P puede ser realizado en un tiempo de orden $O(n\log(n))$ empleando alguno de los algoritmos óptimos de ordenamiento como "heap sort" o "merge sort" [44].

Una vez garantizado que el conjunto generador se encuentra ordenado, el algoritmo es el siguiente:

Algoritmo Divide y Vencerás

1. Si $n \leq 3$, con n el número de puntos generadores, se obtiene el teselado de Voronoi de manera directa (por ejemplo, aplicando el Método Ingenuo) y se entrega al usuario dicho teselado de Voronoi; caso contrario se procede con los siguientes pasos.

2. Sea t la parte entera del número $n/2$. Se divide al conjunto generador en los conjuntos $P_L = \{\mathbf{p}_1, \ldots, \mathbf{p}_t\}$ y $P_R = \{\mathbf{p}_{t+1}, \ldots, \mathbf{p}_n\}$.

3. Se construye el teselado de Voronoi $\mathbb{V}_L$ asociado al conjunto generador P_L aplicando el presente algoritmo.

4. Se construye el teselado de Voronoi $\mathbb{V}_R$ asociado al conjunto generador P_R aplicando el presente algoritmo.

5. Se combinan los teselados de Voronoi $\mathbb{V}_L$ y $\mathbb{V}_R$ en un sólo teselado de Voronoi $\mathbb{V}$ correspondiente al conjunto generador P y se entrega al usuario.

Nótese que, debido a la manera en la que se ordenaron los puntos generadores del conjunto P, los puntos generadores del conjunto P_L se encuentran a la izquierda de los puntos generadores del conjunto P_R. A grandes rasgos este es el algoritmo para el método Divide y Vencerás, sin embargo la parte esencial de dicho algoritmo recae en el paso 5 correspondiente a combinar dos teselados de Voronoi que coexisten en un mismo plano; en los siguientes párrafos nos dedicaremos a detallar cómo se realiza este proceso.

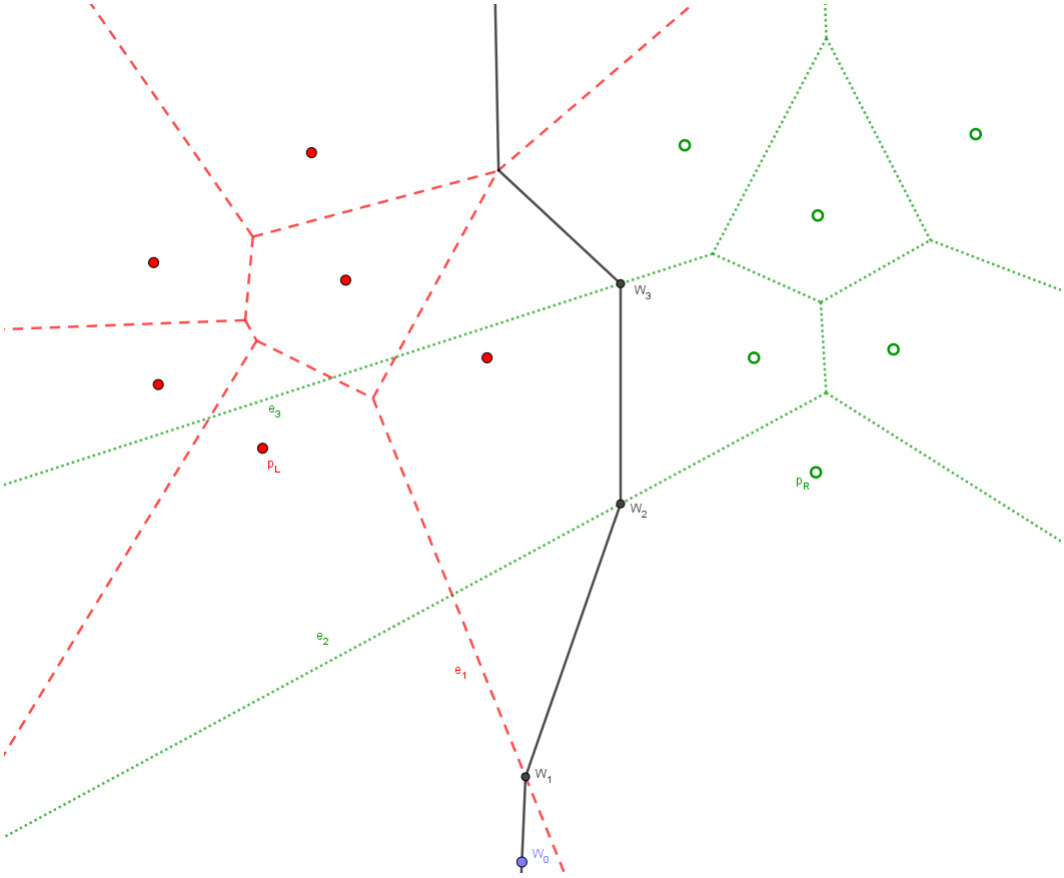

Figura 4.1: Proceso en el que se combinan dos teselados de Voronoi. En color rojo se muestra el teselado de Voronoi asociado a los puntos generadores sólidos y rojos, en color verde se muestra el teselado de Voronoi asociado a los puntos generadores huecos y verdes; en línea negra sólida se muestran los lados de los polígonos de Voronoi que se añaden al combinar ambos teselados de Voronoi. Finalmente se eliminan los segmentos de color rojo a la derecha de la línea negra sólida y los segmentos de color verde a la izquierda de dicha línea.

Supóngase que hemos obtenido los teselados de Voronoi $\mathbb{V}_L$ y $\mathbb{V}_R$ como se muestran en la figura 4.1, donde los puntos generadores del conjunto P_L están indicados por puntos sólidos en color rojo, los puntos generadores del conjunto P_R por puntos huecos de color verde, el teselado $\mathbb{V}_L$ por líneas color rojo y el teselado $\mathbb{V}_R$ por líneas verdes. La cuestión que nos ocupa ahora es unir ambos teselados en uno sólo generando nuevos vértices de Voronoi con sus respectivos bordes, así como eliminar las secciones de los bordes de Voronoi superfluos de cada teselado.

Los nuevos vértices y bordes de Voronoi están determinados por la interacción entre los puntos generadores del conjunto P_L con los puntos generadores del conjunto P_R, de tal forma que cada nuevo borde se construye a partir de la mediatriz asociada a algún segmento que pase por un punto generador en P_L y un punto generador en P_R.

Sea $\mathbf{p} \in \mathbb{R}^2$ un punto arbitrario en el plano y sea $d(\mathbf{p}, P_L)$ la mínima distancia entre el punto $\mathbf{p}$ y los puntos generadores del conjunto P_L, de manera análoga definimos a la distancia $d(\mathbf{p}, P_R)$. Considérese una línea horizontal arbitraria, dado que los puntos generadores del conjunto P_L se encuentran a la izquierda de los puntos generadores del conjunto P_R, existe un único punto $\mathbf{p}^*$ que pertenece a la línea horizontal tal que $d(\mathbf{p}^*, P_L) = d(\mathbf{p}^*, P_R)$; dado que el punto $\mathbf{p}^*$ satisface la igualdad anterior, aseguramos que dicho punto pertenece a uno de los nuevos bordes de Voronoi.

Conforme la línea horizontal se va desplazando de abajo hacia arriba, la posición del punto $\mathbf{p}^*$ dentro de dicha línea va cambiando, trazando una línea poligonal a su paso, de modo que si logramos construir esta línea poligonal y remover de los teselados $\mathbb{V}_L$ y $\mathbb{V}_R$ los segmentos de los bordes que se encuentran a la derecha y a la izquierda, respectivamente, de dicha línea poligonal, habremos combinado adecuadamente ambos teselados de Voronoi.

Para poder construir la línea poligonal mencionada en el párrafo anterior necesitamos antes presentar un nuevo algoritmo que, dados dos polígonos convexos y ajenos, genere una línea que los toque y que al mismo tiempo todos los puntos de ambos polígonos queden en un mismo lado de dicha línea.

Sea U un polígono convexo definido por la lista cíclica de puntos $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m, \mathbf{u}_1)$ en sentido antihorario. Para el vértice $\mathbf{u}_i$ en U, denotamos por next$[\mathbf{u}_i]$ y prev$[\mathbf{u}_i]$ al siguiente vértice y al vértice previo a $\mathbf{u}_i$ en la lista, respectivamente.

Sean U_L y U_R dos polígonos convexos, con U_L a la izquierda de U_R, esto es que las coordenadas en x de los vértices que definen a U_L son menores a las coordenadas en x de los vértices en U_R. Para un vértice $\mathbf{u} \in U_L$ y un vértice $\mathbf{w} \in U_R$, la línea $L(\mathbf{u}, \mathbf{w})$ que pasa por ellos se llama ***soporte común*** de U_L y U_R si todos los vértices de estos polígonos se encuentran en un mismo lado de la línea o sobre la línea. Los polígonos U_L y U_R admiten dos soportes comunes: El soporte común superior y el soporte común

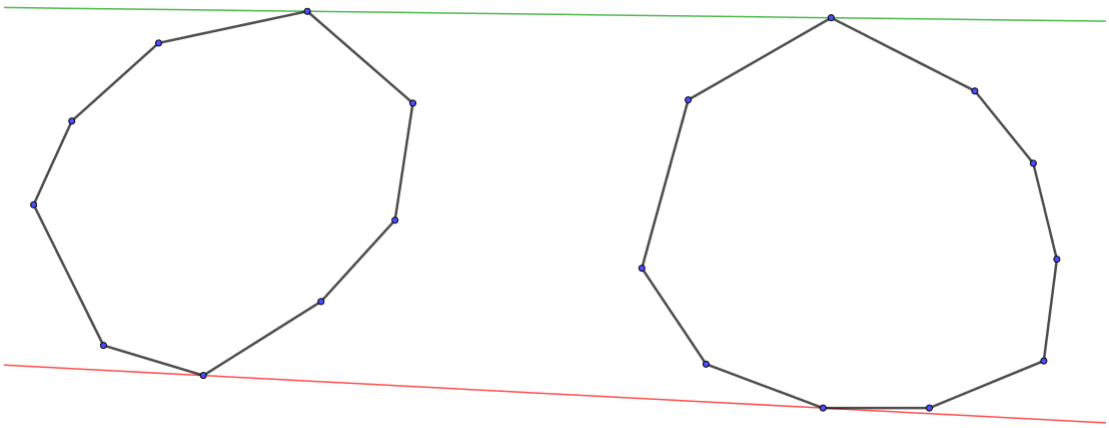

Figura 4.2: Dos polígonos convexos y sus soportes comunes. El soporte común superior está representado por una línea de color verde, mientras que el soporte común inferior lo está por una línea color rojo.

inferior (véase la figura 4.2).

El soporte común inferior puede ser construido a partir del siguiente algoritmo:

Algoritmo Soporte Común Inferior

1. Sea $\mathbf{u} \in U_L$ el vértice con la mayor coordenada en x y sea $\mathbf{w} \in U_R$ el vértice con la menor coordenada en x.
2. Mientras el vértice prev[$\mathbf{u}$] se encuentre por debajo de la línea $L(\mathbf{u}, \mathbf{w})$, se actualiza la variable $\mathbf{u} \leftarrow$ prev[$\mathbf{u}$].
3. Mientras el vértice next[$\mathbf{w}$] se encuentre por debajo de la línea $L(\mathbf{u}, \mathbf{w})$, se actualiza la variable $\mathbf{w} \leftarrow$ next[$\mathbf{w}$].
4. Se iteran los pasos 2 y 3 hasta que los vértices $\mathbf{u}$ y $\mathbf{w}$ no cambien y se regresa al usuario la línea $L(\mathbf{u}, \mathbf{w})$.

Un ejemplo de este algoritmo se muestra en la figura 4.3. En primera instancia el par de puntos $(\mathbf{u}, \mathbf{w})$ obtenido en el paso 1 es $(\mathbf{a}, \mathbf{e})$. Después, el par es cambiado a $(\mathbf{b}, \mathbf{e})$ en el paso 2, en el paso 3 se cambia a $(\mathbf{b}, \mathbf{f})$, se regresa al paso 2 y en dicho paso se cambia al par $(\mathbf{c}, \mathbf{f})$, después al par $(\mathbf{d}, \mathbf{f})$ y finalmente, en el paso 3 se obtiene el par $(\mathbf{d}, \mathbf{g})$.

Sea n el número total de vértices en U_L y U_R. En el paso 1, el par de puntos inicial $(\mathbf{u}, \mathbf{w})$ puede ser encontrado escaneando las listas de vértices. En los pasos 2 y 3, el

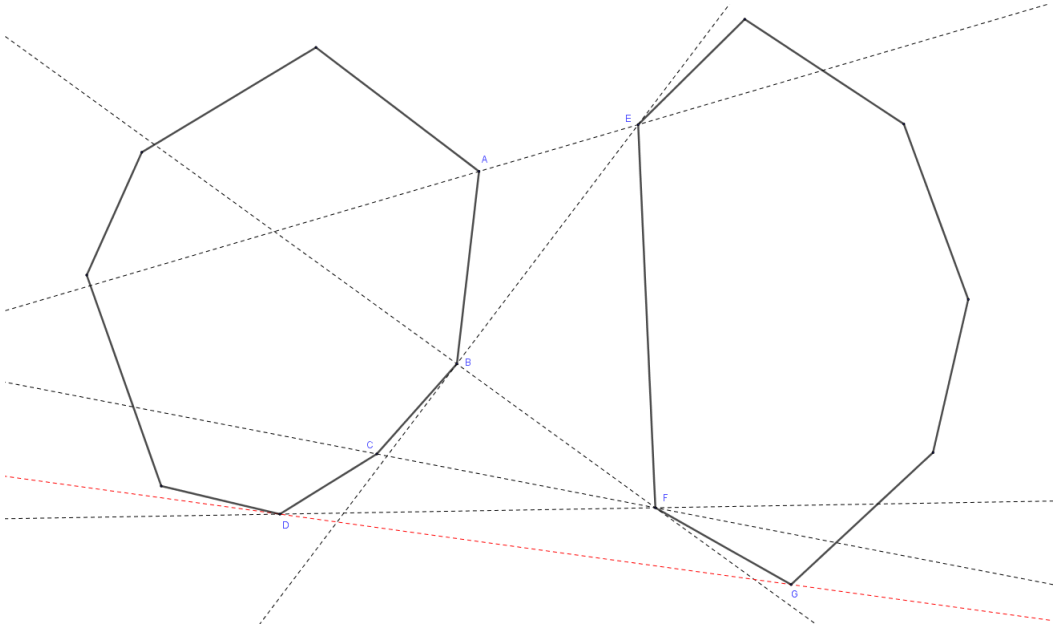

Figura 4.3: Pasos para encontrar el soporte común inferior de un par de polígonos convexos.

número total de actualizaciones del par $(\mathbf{u}, \mathbf{w})$ no supera a n. De esta forma, el algoritmo Soporte Común Inferior requiere un tiempo de orden $O(n)$.

Una vez analizado el algoritmo previo estamos en condiciones de abordar el cómo combinar de manera adecuada dos teselados de Voronoi, esto es, cómo construir la línea poligonal que genera los nuevos vértices y bordes de Voronoi entre ambos teselados. Para explicar a grandes rasgos la idea detrás del algoritmo, apoyémonos en la figura 4.1: Dados los conjuntos generadores P_L y P_R, obtenemos las envolventes convexas de cada uno de dichos conjuntos y las denotamos por U_L y U_R. Sean $\mathbf{p}_L$ y $\mathbf{p}_R$ el par de puntos generadores con los que se construye el soporte común inferior a los polígonos convexos U_L y U_R. Sea $m(\mathbf{p}_L, \mathbf{p}_R)$ la mediatriz al segmento definido por los puntos $\mathbf{p}_L$ y $\mathbf{p}_R$, esta mediatriz nos determina uno de los bordes de Voronoi pues el segmento $\overline{\mathbf{p}_L\mathbf{p}_R}$ es un lado de la envolvente convexa de $P_L \cup P_R$ pero no es un lado de U_L ni de U_R.

Sea $\mathbf{w}_0$ el punto al infinito en la dirección negativa del eje y a lo largo de la mediatriz $m(\mathbf{p}_L, \mathbf{p}_R)$. Consideremos un punto $\mathbf{w}$ que se mueve sobre dicha bisectriz y sea $\mathbf{w}_1$ el punto donde $\mathbf{w}$, viniendo desde $\mathbf{w}_0$, cruza un borde de Voronoi, digamos el borde e_1 (ya sea del teselado $\mathbb{V}_L$ o del teselado $\mathbb{V}_R$) por primera vez. De esta forma, el rayo $\overline{\mathbf{w}_0\mathbf{w}_1}$ es el primer borde de Voronoi que se añade y el punto $\mathbf{w}_1$ el primer vértice de Voronoi que se añade.

Si el borde e_1 pertenece a $\mathbb{V}_L$, reemplazamos el punto $\mathbf{p}_L$ con el punto generador de

P_L que se encuentra al otro lado del borde e_1. Si el borde e_1 pertenece a $\mathbb{V}_R$ (como en el caso de la figura 4.1), reemplazamos el punto $\mathbf{p}_R$ con el punto generador de P_R que se encuentra al otro lado de e_1. Tras cualquiera de los dos casos, obtenemos la mediatriz al segmento que pasa por el nuevo par de puntos $(\mathbf{p}_L, \mathbf{p}_R)$, esta mediatriz pasa por el punto $\mathbf{w}_1$. Supongamos que el punto $\mathbf{w}$ se desplaza a lo largo de esta nueva mediatriz desde el punto $\mathbf{w}_1$ hasta que vuelve a intersectar un borde de Voronoi en el punto $\mathbf{w}_2$. El segmento $\overline{\mathbf{w}_1\mathbf{w}_2}$ es un nuevo borde de Voronoi y el punto $\mathbf{w}_2$ un nuevo vértice de Voronoi. Este procedimiento se itera hasta que el par de punto $(\mathbf{p}_L, \mathbf{p}_R)$ genera el soporte común superior.

Lo descrito en los párrafos anteriores está contenido en el siguiente algoritmo:

Algoritmo Combinación de dos Teselados de Voronoi

1. Se construye la envolvente convexa de cada uno de los conjuntos generadores P_L y P_R, las cuales se denotan como U_L y U_R respectivamente.
2. Se construye el soporte común inferior a los polígonos U_L y U_R a partir del algoritmo Soporte Común Inferior. Dicha recta se denota como $L(\mathbf{p}_L, \mathbf{p}_R)$.
3. Se define al vértice $\mathbf{w}_0$ como el punto al infinito en la dirección negativa del eje y sobre la mediatriz $m(\mathbf{p}_L, \mathbf{p}_R)$ y se fija el índice $i \leftarrow 0$.
4. Se actualiza el índice $i \leftarrow i + 1$.
5. Se calcula el punto de intersección $\mathbf{a}_L$ (que no sea $\mathbf{w}_{i-1}$) entre la mediatriz $m(\mathbf{p}_L, \mathbf{p}_R)$ y los bordes de la celda de Voronoi $V(\mathbf{p}_L)$.
6. Se calcula el punto de intersección $\mathbf{a}_R$ (que no sea $\mathbf{w}_{i-1}$) entre la mediatriz $m(\mathbf{p}_L, \mathbf{p}_R)$ y los bordes de la celda de Voronoi $V(\mathbf{p}_R)$.
7. Si la coordenada en y de $\mathbf{a}_L$ es menor a la coordenada en y de $\mathbf{a}_R$, entonces $\mathbf{w}_i \leftarrow \mathbf{a}_L$ y se redefine al punto $\mathbf{p}_L$ como el punto generador al otro lado del borde de Voronoi que contiene a $\mathbf{a}_L$; en caso contrario $\mathbf{w}_i \leftarrow \mathbf{a}_R$ y se redefine al punto $\mathbf{p}_R$ como el punto generador al otro lado del borde de Voronoi que contiene a $\mathbf{a}_R$.
8. Se iteran los pasos 4-7 hasta que la recta $L(\mathbf{p}_L, \mathbf{p}_R)$ sea el soporte común superior a los polígonos U_L y U_R.
9. Se define la variable $m \leftarrow i$ y al punto $\mathbf{w}_{m+1}$ como el punto al infinito en la dirección positiva del eje y sobre la mediatriz $m(\mathbf{p}_L, \mathbf{p}_R)$.
10. Se construye la línea poligonal $(\overline{\mathbf{w}_0\mathbf{w}_1}, \overline{\mathbf{w}_1\mathbf{w}_2}, \ldots, \overline{\mathbf{w}_m\mathbf{w}_{m+1}})$ y se eliminan los segmentos de los bordes de Voronoi de $\mathbb{V}_L$ que quedan a la derecha de la línea poligonal, así como los segmentos de los bordes de Voronoi de $\mathbb{V}_R$ que quedan a la izquierda de la línea poligonal. Finalmente se regresa al usuario el diagrama de Voronoi final.

El paso 1 requiere un tiempo de orden $O(n)$ dado que ya se tienen los teselados de Voronoi $\mathbb{V}_L$ y $\mathbb{V}_R$. El paso 2 se realiza en un tiempo de orden $O(n)$. El paso 3 se realiza en un tiempo constante. Los pasos 4-7 se repiten a lo más una cantidad n de veces, pues a lo más se pueden agregar n nuevos bordes de Voronoi. Los pasos 4 y 7 se realizan en un tiempo constante.

Los pasos 5 y 6 requieren de un tiempo proporcional al número de bordes que poseen las celdas de Voronoi $V(\mathbf{p}_L)$ y $V(\mathbf{p}_R)$ respectivamente, el cual no es una constante, sin embargo podemos disminuir el número total de bordes de Voronoi a examinar conforme iteramos estos dos pasos del algoritmo si recorremos los bordes de las celdas de Voronoi $V(\mathbf{p}_L)$ en sentido antihorario y los bordes de las celdas de Voronoi $V(\mathbf{p}_R)$ en sentido horario. Supongamos que, en una cierta iteración de los pasos 4-7 hemos encontrado las intersecciones $\mathbf{a}_L$ y $\mathbf{a}_R$ como en la figura 4.4. Dado que $\mathbf{a}_L$ está por debajo de $\mathbf{a}_R$, el punto $\mathbf{p}_L$ se reemplaza por el punto $\mathbf{p}_L^{'}$, causando que la línea poligonal se desvíe hacia la derecha, de tal modo que cuando busquemos el punto de intersección $\mathbf{a}_R^{'}$ entre la mediatriz $b\left(\mathbf{p}_L^{'}, \mathbf{p}_R\right)$ y los bordes de la celda de Voronoi $V(\mathbf{p}_R)$, no necesitamos considerar los bordes a la izquierda de $\mathbf{a}_R$, con lo que podemos empezar a buscar a partir del borde que contiene al punto $\mathbf{a}_R$ y continuar la búsqueda en sentido horario. De esta manera evitamos el revisar sobre todos los bordes de una celda de Voronoi. De manera similar, si el punto de intersección $\mathbf{a}_R$ está por debajo del punto de intersección $\mathbf{a}_L$, el punto $\mathbf{p}_R$ es reemplazado y la línea poligonal se desviará hacia la izquierda, con lo que, para evitar analizar la intersección de la mediatriz $b\left(\mathbf{p}_L, \mathbf{p}_R^{'}\right)$ con todos los bordes de la celda de Voronoi $V(\mathbf{p}_L)$, comenzamos dicha búsqueda en el borde que contiene al punto $\mathbf{a}_L$ y continuamos en sentido antihorario. Siguiendo este procedimiento, los pasos 5 y 6 se realizan en un tiempo de orden $O(n)$.

El paso 9 se realiza en un tiempo constante y el paso 10 se completa en un tiempo de orden $O(n)$. De esta forma, el algoritmo para combinar dos teselados de Voronoi requiere un tiempo de orden $O(n)$.

Finalmente estamos en condiciones de determinar la eficiencia del algoritmo Divide y Vencerás. Los pasos 1 y 2 se realizan en un tiempo constante y el paso 5 se realiza en un tiempo de orden $O(n)$ aplicando el algoritmo Combinación de dos Teselados de Voronoi. El tiempo $T(n)$ que requiere el algoritmo Divide y Vencerás para un conjunto generador con n elementos está dado por

$$T(n) = 2T\left(\frac{n}{2}\right) + kn,$$

el cual, como analizamos en el caso del algoritmo para intersectar N semiplanos (véase el desarrollo para llegar a la expresión 4.24), implica que $T(n) = O(n\log(n))$.

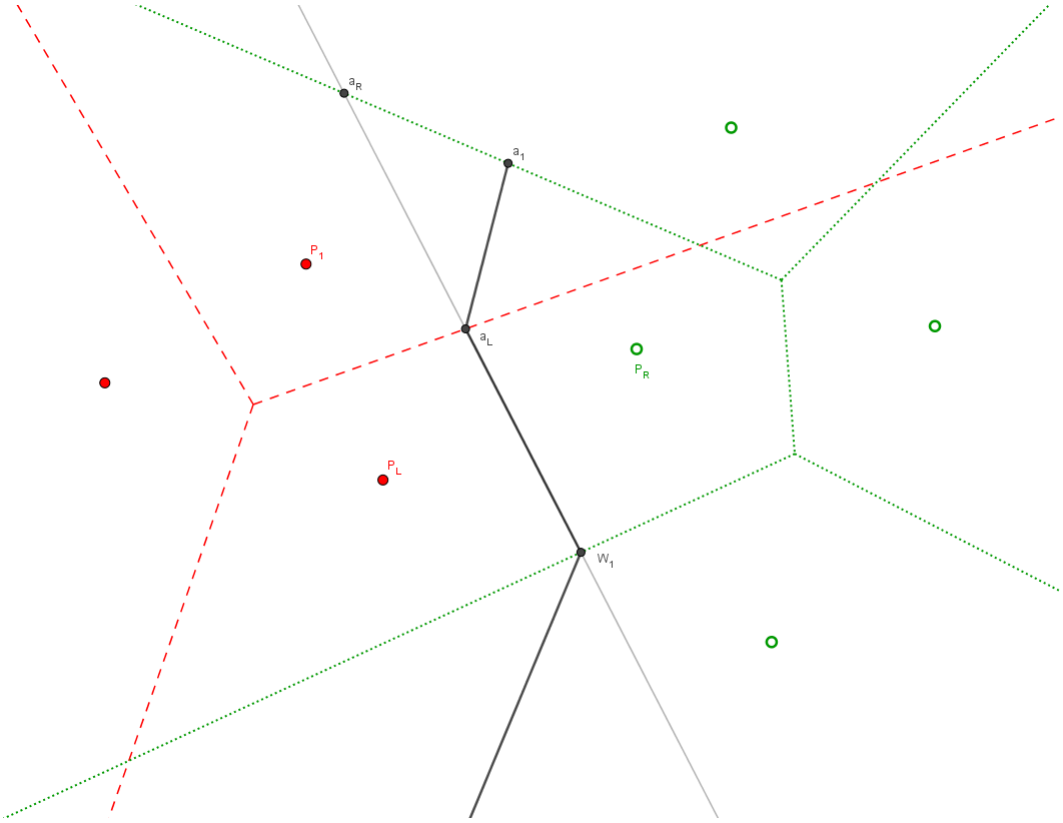

Figura 4.4: Búsqueda del siguiente vértice de Voronoi en los bordes de los teselados de Voronoi $\mathbb{V}_L$ y $\mathbb{V}_R$.

4.3.5. Método Fortune

El presente método está basado en una técnica conocida como "barrido de recta" ("plane sweep" en inglés), la cual consiste en definir una recta vertical (la línea de barrido) y recorrer el plano, de izquierda a derecha, con dicha línea. Conforme la línea de barrido se desplaza, esta tocará a todos los objetos geométricos que estén definidos sobre el plano uno por uno. La idea esencial de la técnica es que, cada que la línea de barrido toque un elemento geométrico, una porción del problema será resuelto.

Una aplicación directa de esta técnica al problema de generar un teselado de Voronoi dado un conjunto generador P no es posible, pues para generar los bordes y vértices de Voronoi asociados a un punto generador $\mathbf{p}_i$ que se encuentre sobre la línea de barrido, es necesario tener información sobre uno o varios puntos generadores que aún no han sido tocados por la línea de barrido previamente; es decir, la línea de barrido toca por primera vez a un polígono de Voronoi antes de que toque a su correspondiente punto generador. No obstante, por medio de un algoritmo propuesto por Steven Fortune, es posible emplear la técnica "barrido de recta" para generar teselados de Voronoi en un tiempo de orden $O(n\log(n))$ [45].

Además de las hipótesis 1, 2 y 3 mencionadas previamente, consideraremos la siguiente hipótesis:

- **Hipótesis 4:** Dado un punto generador $\mathbf{p}_i$ arbitrario, ningún vértice de la celda de Voronoi $V(\mathbf{p}_i)$ está alineado horizontalmente con dicho punto.

Esta hipótesis no es crítica para la implementación del algoritmo, pero vuelve más simple su descripción. A continuación desarrollaremos el mapeo con el cual Fortune fue capaz de implementar la técnica "barrido de recta" al problema de obtener el Teselado de Voronoi dado su conjunto generador $P = \{\mathbf{p}_i, \ldots, \mathbf{p}_n\}$.

Sea $\mathbf{p} \in \mathbb{R}^2$ un punto arbitrario. Definimos al número real $r(\mathbf{p}) = \min\{d(\mathbf{p}, \mathbf{p}_i) \mid 1 \leq i \leq n\}$, donde $d(\mathbf{p}, \mathbf{p}_i)$ representa la distancia entre el punto $\mathbf{p}$ y el punto $\mathbf{p}_i$. El número $r(\mathbf{p})$ es el radio del mayor círculo centrado en $\mathbf{p}$ tal que ningún punto generador queda dentro de él.

Denotando a la coordenada en x del punto $\mathbf{p}$ como $\mathbf{p}_x$ y a la coordenada en y como $\mathbf{p}_y$, definimos al mapeo $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ como

$$\phi(\mathbf{p}) = (\mathbf{p}_x + r(\mathbf{p}), \mathbf{p}_y)\,. \tag{4.25}$$

En la figura 4.5 podemos apreciar de manera gráfica el mapeo 4.25 aplicado a la mediatriz m del segmento formado por los puntos rojos $\mathbf{P}_1$ y $\mathbf{P}_2$. Cada punto $\mathbf{a}_i$ en azul sobre la mediatriz es el centro de un círculo de radio $r(\mathbf{a}_i)$ (indicado por la línea punteada) y el punto al final de cada flecha que parte de $\mathbf{a}_i$ es $\phi(\mathbf{a}_i)$. La curva negra

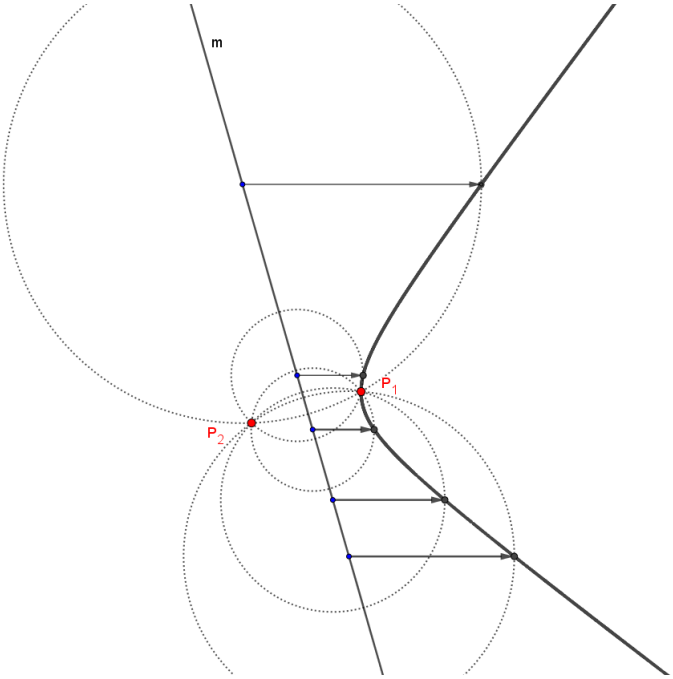

Figura 4.5: Representación gráfica del mapeo 4.25 aplicado a los puntos de la mediatriz m del segmento formado por los puntos rojos $\mathbf{P}_1$ y $\mathbf{P}_2$.

remarcada corresponde al mapeo aplicado a cada punto de la mediatriz m.

Denotemos a los vértices de Voronoi por $\mathbf{q}$, a los bordes de Voronoi por e, a los polígonos de Voronoi por $V(\mathbf{p}_i)$ y al teselado de Voronoi por $\mathbb{V}$; a los conjuntos que resultan de aplicar el mapeo 4.25 a cada uno de estos conjuntos los denotaremos como $\phi(\mathbf{q}), \phi(e), \phi(V(\mathbf{p}_i))$ y $\phi(\mathbb{V})$ respectivamente.

El conjunto $\phi(\mathbb{V})$ puede ser interpretado como un teselado de Voronoi en un río [46] para el caso particular en el que un bote se mueve a la misma velocidad que el cauce del río. Supongamos que los puntos generadores $\mathbf{p}_i$ son islas puntuales localizadas dentro de un gran río tal que cada una de ellas posee un bote que se desplaza con una rapidez constante w y consideremos un punto $\mathbf{p} \in V(\mathbf{p}_i)$ arbitrario para algún índice i. Si el río estuviera estático, el barco que parte de la isla $\mathbf{p}_i$ con dirección al punto $\mathbf{p}$ llegaría antes a dicho punto (en un tiempo $t_s = d(\mathbf{p}, \mathbf{p}_i)/w$) que cualquier otro barco que parta de las demás islas con dirección a ese mismo punto.

Supongamos ahora que el río se desplaza de izquierda a derecha (es decir, en la dirección del eje x) con una rapidez constante igual a la rapidez del bote, y el bote parte de la isla $\mathbf{p}_i$ en dirección al punto $\mathbf{p}$. La velocidad efectiva $\mathbf{V}$ del bote será la

suma de la velocidad inicial del bote con la velocidad del río, esto es

$$\mathbf{V}_x = \mathbf{V}_{x_0} + w, \quad \mathbf{V}_y = \mathbf{V}_{y_0},$$

de tal forma que, tras un tiempo t_s el bote se ha desplazado a la posición

$$x = \mathbf{V}_x t_s = (\mathbf{V}_{x_0} + w)\, t_s = \mathbf{V}_{x_0} t_s + w t_s, \tag{4.26}$$

$$y = \mathbf{V}_{y_0} t_s, \tag{4.27}$$

pero tenemos que $\mathbf{V}_{x_0} t_s = \mathbf{p}_x$ y $\mathbf{V}_{y_0} t_s = \mathbf{p}_y$ pues esto corresponde al caso en que el río es estático, sustituyendo estas igualdades así como la expresión para t_s en 4.26 y 4.27 tenemos que

$$x = \mathbf{p}_x + w\left(\frac{d(\mathbf{p}, \mathbf{p}_i)}{w}\right) = \mathbf{p}_x + d(\mathbf{p}, \mathbf{p}_i),$$

$$y = \mathbf{p}_y,$$

que son justamente las componentes del mapeo $\phi(\mathbf{p})$. De esta forma tenemos que $\phi(\mathbf{p})$ representa el punto alcanzado por el bote que parte de la isla más cercana al punto $\mathbf{p}$ con dirección a $\mathbf{p}$; así el mapeo $\phi(\mathbb{V})$ puede ser interpretado como una teselación de la superficie del río en territorios asociadas a cada isla de modo que cualquier punto en el territorio asociado a una isla arbitraria $\mathbf{p}_i$ puede ser alcanzado por un bote que parte de la isla $\mathbf{p}_i$ en un tiempo menor que el que le tomaría a otro bote que partiera de cualquier otra isla (en el caso en que el punto de interés es inaccesible para los botes que parten de alguna determinada isla, el tiempo se considera infinito).

Un ejemplo del mapeo 4.25 aplicado a un teselado de Voronoi se observa en la figura 4.6: En la figura **A)** se muestra el teselado de Voronoi $\mathbb{V}$ asociado al conjunto generador $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_5\}$ mientras que en la figura **B)** se observa el conjunto $\phi(\mathbb{V})$.

Nótese que para todo $i \neq 1$, el punto generador $\mathbf{p}_i$ es el punto más a la izquierda de la región $\phi(V(\mathbf{p}_i))$. Las imágenes de todos los bordes y vértices de Voronoi asociados al punto generador $\mathbf{p}_i$ se encuentran a la derecha de este punto. Es esta característica la que permite aplicar la técnica "barrido de recta" para construir el conjunto $\phi(\mathbb{V})$.

Por el resto de la sección emplearemos la nomenclatura de los teselados de Voronoi para referirnos a las imágenes de los elementos de un teselado de Voronoi $\mathbb{V}$ tras aplicar el mapeo 4.25; de esta manera nos referiremos, por ejemplo, a la imagen de un vértice de Voronoi en $\mathbb{V}$ como un vértice de Voronoi en $\phi(\mathbb{V})$, a la imagen de un borde de Voronoi en $\mathbb{V}$ como un borde de Voronoi en $\phi(\mathbb{V})$, etc.

Cabe mencionar en este punto del texto que la hipótesis 4 lo que nos garantiza en el fondo es que la imagen de cualquier vértice de Voronoi $\phi(\mathbf{q})$ no resulta ser el punto

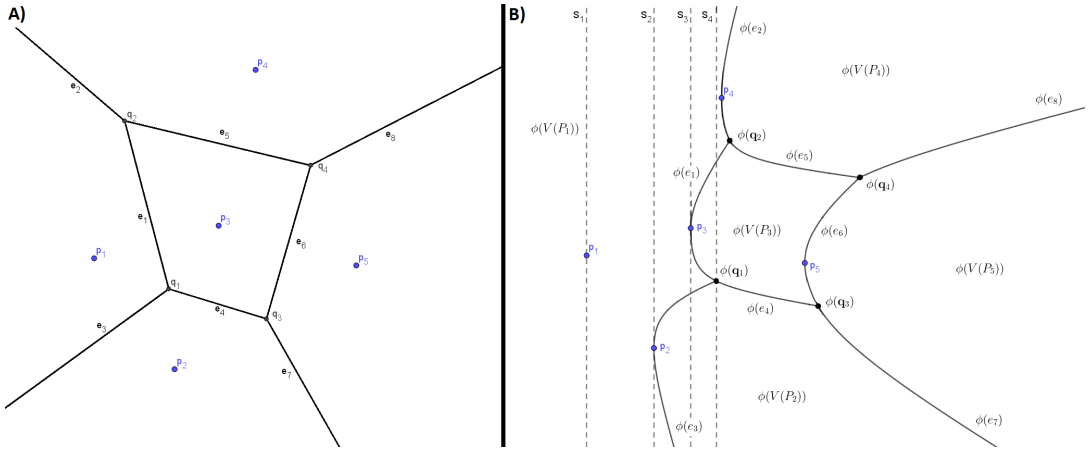

Figura 4.6: A) Teselado de Voronoi $\mathbb{V}$ para un conjunto conjunto P con cinco puntos. **B)** Imagen del mapeo ϕ aplicado al teselado de Voronoi $\mathbb{V}$.

más a la izquierda de alguna región de Voronoi en $\phi(\mathbb{V})$ y, por consiguiente, de cada punto $\phi(\mathbf{q})$ sale un vértice de Voronoi $V(e_j)$ a la derecha y dos vértices de Voronoi $V(e_k)$, $V(e_m)$ a la izquierda para algunos índices j, k, m. Por ejemplo, en la figura **B)** de 4.6, del vértice de Voronoi $\phi(\mathbf{q}_4)$ sale a la derecha el borde de Voronoi $\phi(e_8)$ y a la izquierda los bordes de Voronoi $\phi(e_5)$ y $\phi(e_6)$.

Durante el desarrollo del algoritmo, necesitaremos conocer qué regiones o bordes de Voronoi atraviesa la línea de barrido conforme esta se desplaza por el plano, esta información la guardaremos en una lista ordenada L en donde las regiones y bordes serán almacenadas según el orden en que se intersectan por la línea de barrido de abajo hacia arriba. De igual manera necesitaremos definir un conjunto Q con los puntos potenciales a ser los siguientes eventos que ocurrirán conforme la línea de barrido se desplaza. En un principio este conjunto Q estará conformado únicamente por los puntos generadores, sin embargo conforme el algoritmo avance se irán añadiendo a esta lista los vértices de Voronoi en $\phi(\mathbb{V})$.

La técnica "barrido de recta" moverá de izquierda a derecha una recta vertical a lo largo de todo el plano y, cada que un "evento" ocurra, la lista L y el conjunto Q en la posición actual de la línea de barrido se actualizarán. Existen dos tipos de eventos: el primero de ellos ocurre cuando la línea de barrido toca un punto generador, como es el caso de las líneas S_1, S_2 y S_3 de la figura 4.6 **B)**, el segundo de estos eventos corresponde al caso en el que la línea de barrido toca un vértice de Voronoi de $\phi(\mathbb{V})$, como en la línea S_4 de la figura mencionada previamente.

Sean $\mathbf{p}_i$ y $\mathbf{p}_j$ dos puntos generadores, la mediatriz al segmento cuyos extremos están dados por este par de puntos se transforma, bajo el mapeo ϕ, en una hipérbola. Dado el punto más a la izquierda de dicha hipérbola, definimos como $h^+(\mathbf{p}_i, \mathbf{p}_j)$ y

$h^-(\mathbf{p}_i, \mathbf{p}_j)$ a la parte superior e inferior, respectivamente, de la hipérbola. Nótese que $h^+(\mathbf{p}_i, \mathbf{p}_j) = h^+(\mathbf{p}_j, \mathbf{p}_i)$ y $h^-(\mathbf{p}_i, \mathbf{p}_j) = h^-(\mathbf{p}_j, \mathbf{p}_i)$.

Una vez mencionado todo lo anterior, procedemos a presentar el algoritmo de Fortune propiamente dicho:

Algoritmo Fortune

1. Se define el conjunto $Q \leftarrow P$, con P el conjunto generador.
2. Se selecciona y se elimina de Q el punto generador $\mathbf{p}_i$ más a la izquierda en el plano.
3. Se define la lista L, cuyo único elemento es la región $\phi(V(\mathbf{p}_i))$.
4. Mientras Q sea un conjunto no vacío, se iterarán los pasos 4.a), 4.b) y 4.c)
   - *a*) Se selecciona y se elimina, del conjunto Q, el punto $\mathbf{w}$ más a la izquierda en el plano.
   - *b*) Si $\mathbf{w}$ es un punto generador, digamos $\mathbf{w} = \mathbf{p}_j$, se realizan los pasos 4.b.1), 4.b.2) y 4.b.3)
     1) Se encuentra la región $\phi(V(\mathbf{p}_k))$ en L que contiene al punto $\mathbf{p}_j$.
     2) Se reemplaza el elemento $\phi(V(\mathbf{p}_k))$ en L por la secuencia de elementos $(\phi(V(\mathbf{p}_k)), h^-(\mathbf{p}_k, \mathbf{p}_j), \phi(V(\mathbf{p}_j)), h^+(\mathbf{p}_j, \mathbf{p}_k), \phi(V(\mathbf{p}_k)))$.
     3) Se añade al conjunto Q el punto de intersección entre $h^-(\mathbf{p}_k, \mathbf{p}_j)$ con la parte superior de la hipérbola inmediatamente debajo que aparece en la lista L, así como el punto de intersección entre $h^+(\mathbf{p}_j, \mathbf{p}_k)$ con la parte inferior de la hipérbola inmediatamente por encima que aparece en la lista L.
   - *c*) Si $\mathbf{w} = \phi(\mathbf{q})$ es un punto de intersección entre $h^{\pm}(\mathbf{p}_j, \mathbf{p}_k)$ y $h^{\mp}(\mathbf{p}_k, \mathbf{p}_m)$ para algunos índices j, k y m, se realizan los pasos 4.c.1), 4.c.2), 4.c.3) y 4.c.4).
     1) Se reemplaza la secuencia $(h^{\pm}(\mathbf{p}_j, \mathbf{p}_k), \phi(V(\mathbf{p}_k)), h^{\pm}(\mathbf{p}_k, \mathbf{p}_m))$ en la lista L por el elemento $h = h^-(\mathbf{p}_j, \mathbf{p}_m)$ o el elemento $h = h^+(\mathbf{p}_j, \mathbf{p}_m)$ dependiendo de cuál de ellos contiene al punto de intersección.
     2) Se elimina del conjunto Q cualquier intersección de $h^{\pm}(\mathbf{p}_j, \mathbf{p}_k)$ o $h^{\mp}(\mathbf{p}_k, \mathbf{p}_m)$ con cualquier otra hipérbola.
     3) Se añade al conjunto Q cualquier intersección de h con la parte superior de la hipérbola inmediatamente por encima o con la parte inferior de la hipérbola inmediatamente por debajo de h en la lista L.
     4) Se marca al punto $\phi(\mathbf{q})$ como un vértice de Voronoi en $\phi(\mathbb{V})$ incidente a $h^{\pm}(\mathbf{p}_j, \mathbf{p}_k), h^{\mp}(\mathbf{p}_k, \mathbf{p}_m)$ y h.
5. Se regresa al usuario la lista L, los puntos de intersección marcados $\phi(\mathbf{q}_i)$ y la relación de incidencia entre los elementos de ambos conjuntos.

Los pasos 1, 2 y 3 son los que dan inicio al algoritmo. En el caso particular que se muestra en la imagen 4.6 **B)** el conjunto Q se define, durante el paso 1, como $Q = \{\mathbf{p}_1, \ldots, \mathbf{p}_5\}$; en el paso 2 se selecciona el punto $\mathbf{p}_i = \mathbf{p}_1$ y el conjunto Q se redefine como $Q = \{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$; en el paso 3 la lista L se define como $L = (\phi(V(\mathbf{p}_1)))$. Los pasos 2 y 3 corresponden al caso en que la línea de barrido se encuentra en la posición de la línea vertical S_1 tocando al punto generador $\mathbf{p}_1$. La lista $L = (\phi(V(\mathbf{p}_1)))$ nos indica que, estando la línea de barrido a una distancia infinitesimal a la derecha de la línea S_1, si nos movemos a lo largo de la línea de barrido de abajo hacia arriba, siempre estaremos en la región de Voronoi asociada al punto $\mathbf{p}_1$.

El paso 4 es la rutina central del algoritmo. En el paso 4.a) elegimos el próximo punto que desencadena un evento al ser tocado por la línea de barrido. En el ejemplo que estamos analizando, el algoritmo se desarrolla de la siguiente forma: al final del paso 3, el conjunto Q quedó como $Q = \{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$ de tal forma que durante el paso 4.a) se selecciona al punto $\mathbf{w} = \mathbf{p}_2$ y se redefine a Q como $Q = \{\mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$. Esto corresponde al evento desencadenado cuando la línea de barrido toca al punto generador $\mathbf{p}_2$, es decir, cuando la línea de barrido está en la posición de la recta S_2.

Dado que el punto $\mathbf{w} = \mathbf{p}_2$ es un punto generador, el algoritmo entra al paso 4.b). Para ello, en el paso 4.b.1) encontramos la región $\phi(V(\mathbf{p}_k))$ que contiene al punto $\mathbf{w}$, esta región es para nuestro caso particular $\phi(V(\mathbf{p}_1))$; en el paso 4.b.2) la lista L es redefinida como

$$L = \big(\phi(V(\mathbf{p}_1)), h^-(\mathbf{p}_1, \mathbf{p}_2), \phi(V(\mathbf{p}_2)), h^+(\mathbf{p}_2, \mathbf{p}_1), \phi(V(\mathbf{p}_1))\big)\,,$$

donde lo que nos indica ahora es que, si la línea de barrido se encuentra a una distancia infinitesimal a la derecha de la línea S_2 y nos movemos a lo largo de la línea de barrido de abajo hacia arriba, primero estaremos en la región $\phi(V(\mathbf{p}_1))$, después cruzaremos el borde $h^-(\mathbf{p}_1, \mathbf{p}_2)$ para ingresar a la región $\phi(V(\mathbf{p}_2))$, la cual dejaremos tras cruzar el borde $h^+(\mathbf{p}_2, \mathbf{p}_1)$ y finalmente estaremos en la región $\phi(V(\mathbf{p}_1))$ nuevamente. Dado que no existe ningún otra hipérbola más que la formada por $h^-(\mathbf{p}_1, \mathbf{p}_2)$ y $h^+(\mathbf{p}_2, \mathbf{p}_1)$, en el paso 4.b.3) no se realiza nada.

En la siguiente iteración del paso 4, durante el paso 4.a), el punto $\mathbf{w} = \mathbf{p}_3$ es seleccionado y el conjunto Q redefinido como $Q = \{\mathbf{p}_4, \mathbf{p}_5\}$, lo anterior corresponde a la situación en la que la línea de barrido se encuentra en la posición de la línea S_3. Dado que el punto $\mathbf{w} = \mathbf{p}_3$ es un punto generador, el algoritmo entra en el paso 4.b) en donde, dada la lista L, la región que contiene al punto $\mathbf{w}$ es $\phi(V(\mathbf{p}_1))$. En el paso 4.b.2) se redefine la lista L quedando de la siguiente manera

$$L = \big(\phi(V(\mathbf{p}_1)), h^-(\mathbf{p}_1, \mathbf{p}_2), \phi(V(\mathbf{p}_2)), h^+(\mathbf{p}_2, \mathbf{p}_1),$$

$$\phi(V(\mathbf{p}_1)), h^-(\mathbf{p}_1, \mathbf{p}_3), \phi(V(\mathbf{p}_3)), h^+(\mathbf{p}_3, \mathbf{p}_1), \phi(V(\mathbf{p}_1))\big) .$$

En la lista L se encuentra la semi-hipérbola $h^+(\mathbf{p}_2, \mathbf{p}_1)$ inmediatamente debajo de $h^-(\mathbf{p}_1, \mathbf{p}_3)$, de tal modo que en el paso 4.b.3), el punto $\phi(\mathbf{q}_1)$ resultado de la intersección de las semi-hipérbolas mencionadas anteriormente es agregado al conjunto Q, con lo que este conjunto termina siendo $Q = \{\phi(\mathbf{q}_1), \mathbf{p}_4, \mathbf{p}_5\}$.

En la tercera iteración del paso 4, durante el paso 4.a), el punto $\mathbf{w} = \phi(\mathbf{q}_1)$ es seleccionado y el conjunto Q redefinido como $Q = \{\mathbf{p}_4, \mathbf{p}_5\}$, esto corresponde a la situación en la que la línea de barrido se encuentra en la posición de la línea S_4. Dado que el punto $\mathbf{w} = \phi(\mathbf{q}_1)$ es el punto de intersección entre las semi-hipérbolas $h^-(\mathbf{p}_3, \mathbf{p}_1)$ y $h^+(\mathbf{p}_1, \mathbf{p}_2)$, el algoritmo entra en el paso 4.c) en donde, durante el paso 4.c.1), se reemplaza la secuencia $(h^-(\mathbf{p}_3, \mathbf{p}_1), \phi(V(\mathbf{p}_1), h^+(\mathbf{p}_1, \mathbf{p}_2)))$ por el elemento $h^-(\mathbf{p}_2, \mathbf{p}_3)$, quedando la lista L como

$$L = \{\phi(V(\mathbf{p}_1)), h^-(\mathbf{p}_1, \mathbf{p}_2), \phi(V(\mathbf{p}_2)), h^-(\mathbf{p}_2, \mathbf{p}_3), \phi(V(\mathbf{p}_3)), h^+(\mathbf{p}_3, \mathbf{p}_1), \phi(V(\mathbf{p}_1))\}.$$

El paso 4.c.2) no se realiza pues el conjunto Q no contiene otro punto de intersección entre $h^-(\mathbf{p}_3, \mathbf{p}_1)$ y/o $h^+(\mathbf{p}_1, \mathbf{p}_2)$ con algún otra hipérbola en el plano. De igual forma el paso 4.c.3) no se realiza pues la semi-hipérbola $h^-(\mathbf{p}_2, \mathbf{p}_3)$ no se intersecta con $h^-(\mathbf{p}_1, \mathbf{p}_2)$ ni con $h^+(\mathbf{p}_3, \mathbf{p}_1)$. En el paso 4.c.4) la relación de incidencia entre el vértice de Voronoi $\phi(\mathbf{q}_1)$ y los bordes y regiones de Voronoi de la lista L se realiza.

El algoritmo continúa iterando el paso 4 hasta que el conjunto Q quede vacío, tratando cada evento de manera análoga a lo desarrollado a detalle en párrafos anteriores. El teselado de Voronoi $\phi(\mathbb{V})$ se obtiene de considerar todas las semi-hipérbolas contenidas en la lista final L, todos los vértices de Voronoi $\phi(\mathbf{q})$ marcados y sus relaciones entre ambos elementos. Para construir el teselado de Voronoi $\mathbb{V}$ usual nos basta con aplicar el mapeo inverso, el cual existe debido a que el mapeo 4.25 es un mapeo uno a uno [45].

La eficiencia de este algoritmo depende fuertemente de la estructura de datos que se utilice para el conjunto Q y la lista ordenada L. Las operaciones básicas que serán llevadas a cabo sobre el conjunto Q son el añadir nuevos elementos y eliminar aquel elemento que posea la menor coordenada en x, es decir, el orden en el que los elementos se agregan al conjunto no determina el orden en que se eliminan del mismo; al tipo abstracto de datos que cumplen con esta propiedad se les conoce como "colas de prioridad" y, empleando una estructura de árbol binario conocida como "heap", estas operaciones pueden ser realizadas en un tiempo de orden $O(\log(n))$ con n el número de elementos que posee Q [40]. Durante el paso 4.c.2) se requiere eliminar de Q aquellos elementos que corresponden a los puntos de intersección entre dos semi-hipérbolas, estos puntos no necesariamente cumplen la condición de tener la menor coordenada en x conforme se van eliminando, sin embargo es posible acceder a estos elementos en Q si genera-

mos punteros que los relacionen con las semi-hipérbolas que dan lugar a cada punto, de modo que pueden ser eliminados del conjunto Q en un tiempo de orden $O(\log(n))$ de la misma manera que cuando se elimina al elemento cuya coordenada en x sea menor.

El otro conjunto de datos importante L es una lista ordenada cuyos elementos son, de manera alternada, regiones del plano y semi-hipérbolas a lo largo de la línea de barrido. La intersección de la línea de barrido con las regiones del plano que aparecen en L nos da como resultado intervalos sobre la línea de barrido cuyas fronteras son las intersecciones de la línea de barrido con las semi-hipérbolas. Los intervalos que corresponden a cada región del plano $\phi(V(\mathbf{p}_i))$ cambian conforme la línea de barrido se desplaza. Las operaciones básicas a llevarse a cabo sobre L son la búsqueda de algún intervalo en particular (paso 4.b.2), así como el añadir y eliminar secuencias de la lista ordenada (pasos 4.b.2 y 4.b.3). Una estructura de datos adecuada para realizar estos pasos es conocida como "diccionario" y en nuestro caso particular se requiere de un árbol binario o ternario para implementarse. Con esta estructura de datos, las operaciones requeridas sobre L pueden realizarse en un tiempo de orden $O(\log(n))$.

Implementando adecuadamente las estructuras de datos mencionadas en párrafos anteriores, el algoritmo Fortune requiere un tiempo de orden $O(n\log(n))$ en ser ejecutado. Notemos que el paso 2, por lo manifestado dos párrafos antes, se realiza en un tiempo $O(\log(n))$ mientras que el paso 3 se realiza en un tiempo constante al ser una lista con un único elemento. Considérese que el número máximo de elementos que tendrá la lista L es de orden n, esto debido a que el número de vértices y bordes de Voronoi es, en ambos casos, de orden n. De igual forma, el número máximo de elementos que tendrá el conjunto Q es de orden n puesto que de dicho orden es la cantidad de parejas de bordes de Voronoi que son vecinos en la lista L, además de que los candidatos a vértices de Voronoi son igualmente de orden n.

Con base en lo expuesto en párrafos anteriores, los pasos 4.a), 4.b) y 4.c) requieren un tiempo de orden $O(\log(n))$ cada uno de ellos, mientras que el paso 4 en su totalidad se iterará una cantidad de pasos de orden n, de tal forma que el algoritmo Fortune presenta en su totalidad una complejidad computacional de orden $O(n\log(n))$.

Capítulo 5

Código y detalles de la simulación

Una de las principales desventajas que presenta el método generalizado dual recae en el hecho de que las retículas cuasiperiódicas generadas por una implementación directa de dicho método están limitadas a ser una vecindad alrededor del origen del sistema de referencia empleado. Lo anterior implica que si estamos interesados en estudiar un sistema con simetría rotacional M lejos del origen, digamos a una distancia R, la complejidad del algoritmo es $\sim O(R^2M^2)$, aunado a esto tenemos el problema de la memoria requerida para guardar los datos generados, la mayoría de los cuales no serán de nuestro interés.

En esta sección revisaremos la estructura del código desarrollado en el lenguaje de programación Julia para generar vecindades de un arreglo cuasiperiódico alrededor de un punto arbitrario en el plano, además de un par de aplicaciones de dicho algoritmo en las cuales calculamos el desplazamiento cuadrático medio de estos sistemas cuando el potencial de interacción entre una partícula y los sitios del arreglo es un potencial de amarre fuerte, así como la distribución de vuelos libres en el caso particular de un potencial de esferas duras. En el presente contexto, entiéndase por un potencial de amarre fuerte aquel en el cual la partícula únicamente se ve afectada por el sitio más cercano a ella del arreglo cuasiperiódico, por ejemplo un potencial de esferas duras.

De igual forma expondremos de manera breve y concisa cada una de las funciones que conforman al código con el objeto de que cualquier uso o modificación posterior sea lo más sencillo posible, es por ello que esta exposición será heurística dejando los detalles más técnicos (como el pseudocódigo de los algoritmos) para un apéndice a este trabajo.

5.1. Estructura del código

El código desarrollado se divide en dos secciones: la primera de ellas corresponde a la implementación del algoritmo que genera la vecindad de un arreglo cuasiperiódico

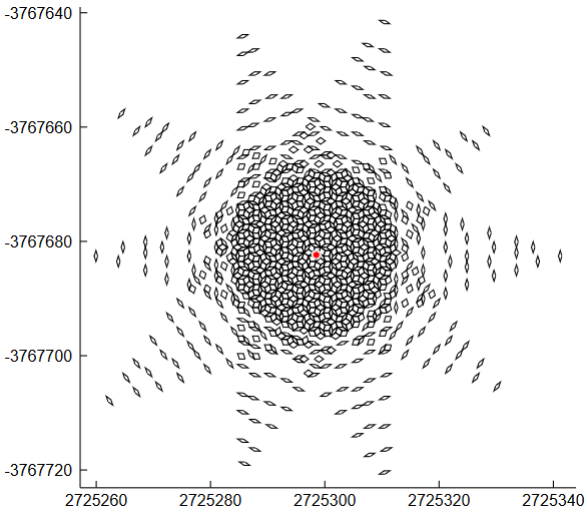

Figura 5.1: Conjunto de polígonos de un arreglo cuasiperiódico con simetría pentagonal, $N = 5$, generado alrededor del punto rojo $\mathbf{P} = (2725298.502632678, -3767682.37123011)$ con las constantes $\alpha_i = 0.2$ y $\beta_i = 5\ \forall i \in \{1, 2, \ldots, 5\}$.

alrededor de un punto arbitrario en el plano, la segunda sección engloba a las funciones y algoritmos necesarios para calcular el desplazamiento cuadrático medio de una partícula dentro de un arreglo cuasiperiódico considerando un potencial de interacción de amarre fuerte entre una partícula y los sitios del arreglo. A partir de ligeras modificaciones a algunas de las funciones contenidas en la segunda sección para el caso particular de un potencial de esferas duras, se puede calcular la distribución de vuelos libres de una partícula en dicho sistema; dichas modificaciones serán comentadas durante la presentación de las funciones correspondientes.

5.1.1. Vecindades del Arreglo Cuasiperiódico

Basados en las expresiones analíticas 3.6 - 3.9 para los vértices de los polígonos que conforman al arreglo cuasiperiódico, así como en las expresiones 3.10 y 3.11 para aproximar al número entero n_i asociado al vector estrella $\mathbf{e}_i$ que requieren las expresiones previamente mencionadas, podemos obtener un conjunto de polígonos del arreglo cuasiperiódico alrededor de un punto arbitrario $\mathbf{P} \in \mathbb{R}^2$. Se considerará un mismo margen de error β_i asociado a los números enteros n_i correspondientes a cada uno de los vectores estrella $\mathbf{e}_i$, así como una misma constante α_i; esto es $\beta_i = \beta,\ \alpha_i = \alpha,\ \forall i$. Véase la figura 5.1.

Este conjunto de polígonos estará conformado en general por varios clústers de polígonos, así como varios polígonos aislados, lo cual representa un problema respecto a la memoria en la computadora que requerimos para almacenar estos datos ya que únicamente son de interés aquellos polígonos que conforman el clúster central (aquel

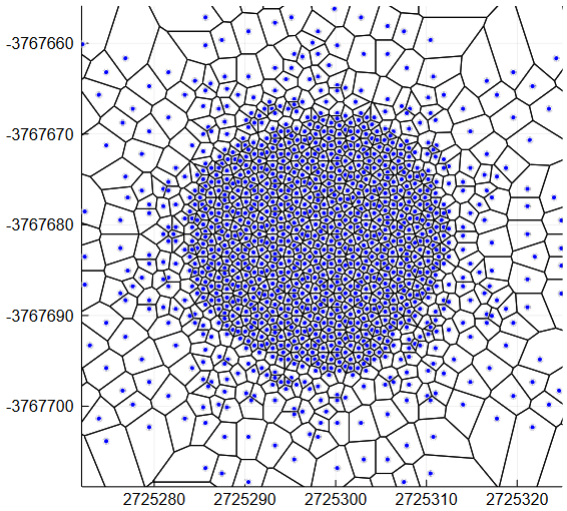

Figura 5.2: Teselado de Voronoi asociado al conjunto de centroides (puntos azules) de los polígonos del arreglo cuasiperiódico. Nótese la diferencia de áreas entre las celdas de Voronoi de los centroides de los polígonos que conforman el clúster central y las celdas de Voronoi de los centroides aislados y/o fronteras de los clústers.

subconjunto de polígonos que están conectados unos con otros y que cumple la condición de que al menos uno de dichos polígonos contiene al punto $\mathbf{P}$).

Considerando el teselado de Voronoi que se forma al tomar como centros de Voronoi a los centroides de estos polígonos, observamos que las celdas de Voronoi que se generan a partir de los centroides de los polígonos aislados o que se encuentran en la frontera de los clústers poseen un área mayor que la de las celdas generadas a partir de los centroides de los polígonos que se encuentran dentro de un clúster. Véase la figura 5.2.

Lo anterior obedece al hecho de que el área de las celdas de Voronoi asociadas a los polígonos internos de un clúster está acotada superiormente, mientras que el área de las celdas de Voronoi asociadas a los polígonos aislados o que pertenecen a la frontera de un clúster no lo está (en particular los polígonos más alejados del clúster principal generarán celdas de Voronoi abiertas con un área infinita), de esta forma podemos identificar a los polígonos aislados o que pertenecen a la frontera de algún clúster y eliminarlos del conjunto de polígonos del arreglo cuasiperiódico.

Para ello se genera el teselado de Voronoi con los centroides de los polígonos del arreglo cuasiperiódico que estamos considerando y, dado un número real A, iteramos sobre el área de las celdas de Voronoi: si el área de dicha celda es mayor que A, eliminamos el polígono cuyo centroide es el centro de Voronoi asociado a dicha celda. Iterando un número I_V de veces este algoritmo obtenemos el clúster central (el número

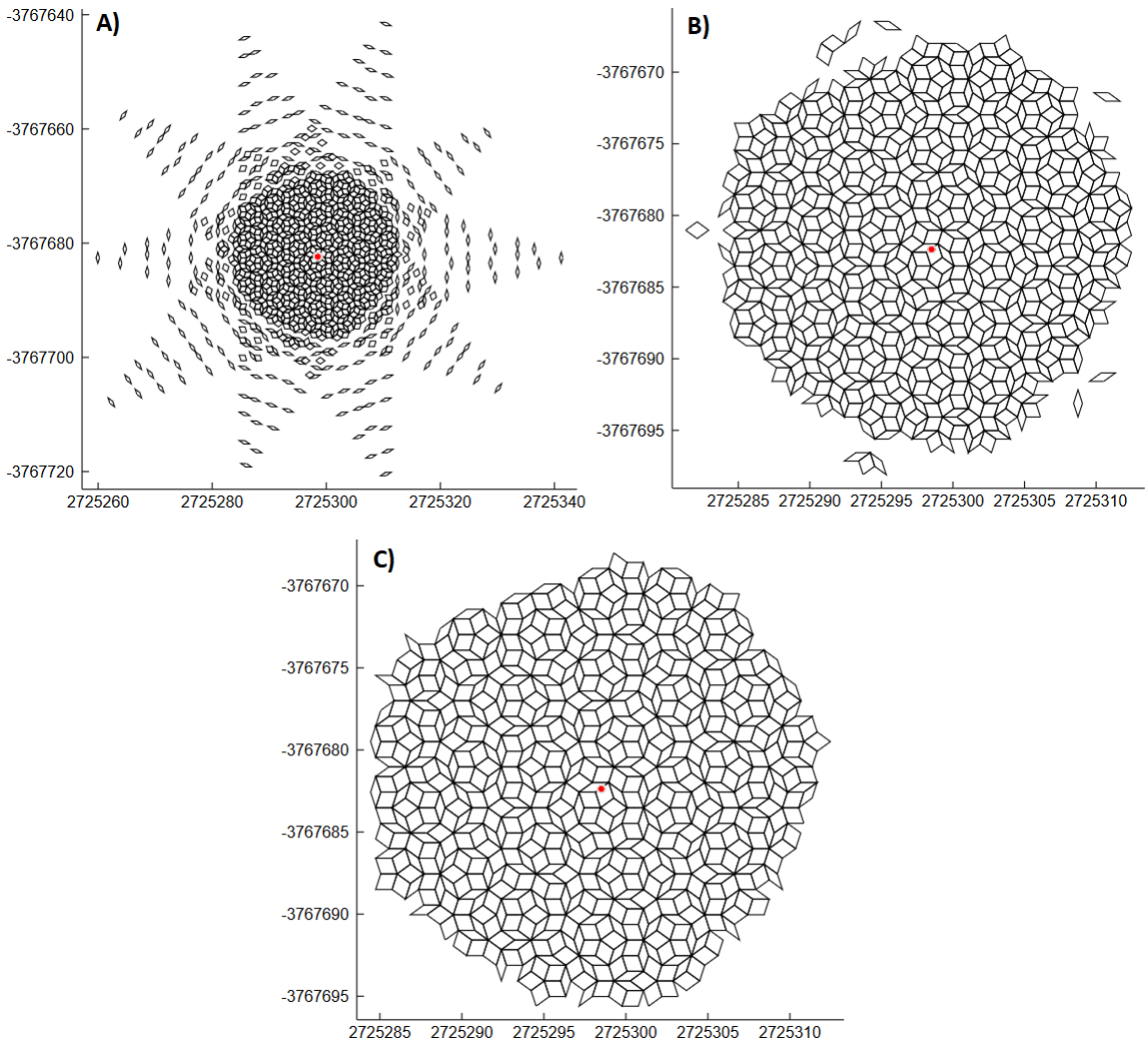

Figura 5.3: A) Conjunto de polígonos de un arreglo cuasiperiódico con simetría pentagonal generado alrededor del punto rojo $\mathbf{P} = (2725298.502632678, -3767682.37123011)$ con las constantes $\alpha_i = 0.2$ y $\beta_i = 5$; **B)** Conjunto de polígonos tras una primera iteración del algoritmo basado en las áreas de las celdas de Voronoi asociadas a los centroides de los polígonos del conjunto en A); **C)** Conjunto de polígonos tras una segunda iteración de dicho algoritmo. En ambas iteraciones se tomó $A = 1.2$ como una cota superior para el área de las celdas.

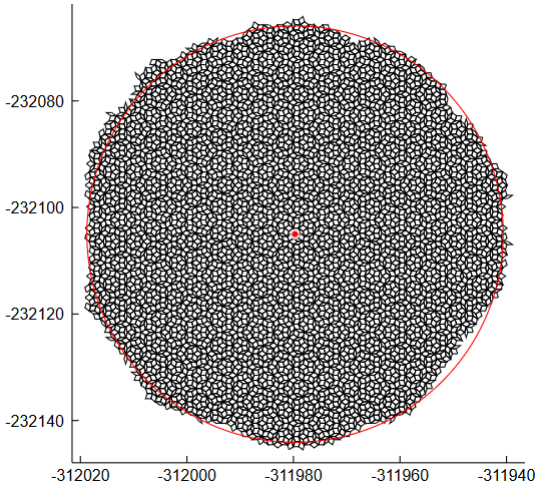

Figura 5.4: Vecindad de un arreglo cuasiperiódico con simetría pentagonal generada alrededor del punto $\mathbf{P} = (-311979.7017913388, -232105.0031628682)$ con una constante $\alpha = 0.2$, un margen de error $\beta = 20$ y un valor de $I_V = 20$. El valor del radio $R_A = 39.0844696836$ se obtuvo analizando 50 vecindades diferentes.

I_V dependerá del tamaño del clúster central, del valor de la cota superior A que se proporcione y de la simetría rotacional del arreglo cuasiperiódico). Véase la figura 5.3.

Una vez obtenido el clúster central del arreglo cuasiperiódico (de ahora en adelante, la vecindad del arreglo cuasiperiódico) tras aplicar I_V veces el algoritmo basado en las áreas de las celdas de Voronoi, aproximamos a dicho clúster por un círculo centrado en el punto $\mathbf{P} \in \mathbb{R}^2$ alrededor del cual generamos nuestro conjunto inicial de polígonos; el que tan buena aproximación resulte dependerá de la simetría rotacional del arreglo cuasiperiódico y del número de iteraciones I_V que se hayan empleado para obtener el clúster central. Véase la figura 5.4.

Fijando la simetría rotacional N del arreglo cuasiperiódico en el que estamos interesados, la constante α, el margen de error β y el número de iteraciones I_V del algoritmo para eliminar polígonos aislados o que pertenecen a la frontera de algún clúster, el radio del círculo R_A que aproxima a las vecindades del arreglo cuasiperiódico generadas por estos parámetros se obtiene a partir del siguiente algoritmo:

1. Se genera de manera aleatoria un punto arbitrario $\mathbf{P} \in \mathbb{R}^2$, alrededor de dicho punto generamos una vecindad del arreglo cuasiperiódico con los parámetros previamente establecidos.

2. Se calcula el siguiente conjunto

$$\{|\mathrm{Max}_x - \mathbf{P}_x|, |\mathrm{Min}_x - \mathbf{P}_x|, |\mathrm{Max}_y - \mathbf{P}_y|, |\mathrm{Min}_y - \mathbf{P}_y|\},$$

donde $\text{Max}_{x(y)}$ es el máximo valor de la coordenada $x(y)$ de todos los vértices de los polígonos que conforman la vecindad del arreglo cuasiperiódico y $\text{Min}_{x(y)}$ es el mínimo valor de la coordenada $x(y)$ de dichos vértices. De este conjunto de cuatro valores se selecciona el menor.

3. Se iteran los pasos previos hasta tener una buena muestra estadística y se calcula el promedio de los valores obtenidos en el inciso dos, dicho promedio corresponde al radio R_A.

Como se puede apreciar en la figura 5.4, esta aproximación no es perfecta y en general habrá regiones dentro del círculo en donde no haya ningún polígono del arreglo cuasiperiódico; para evitar lo anterior definimos un parámetro $\gamma \in (0, 1)$ que escala el valor del radio R_A a fin de que los círculos con el radio escalado satisfagan la condición de que su interior está completamente cubierto por polígonos del arreglo cuasiperiódico. Denotamos como R_S al radio escalado, donde $R_S = \gamma R_A$. El valor del parámetro γ va a depender de qué tan bien se aproximen las vecindades del arreglo cuasiperiódico por el círculo con radio R_A.

A partir de esta aproximación por medio de un círculo de radio R_S a las vecindades del arreglo cuasiperiódico, podemos producir una vecindad de mayor tamaño, la cual sabremos que contiene una región cuadrada de lado $L_S = \sqrt{8}R_S$ con la característica de que dicha región estará completamente cubierta por polígonos del arreglo cuasiperiódico.

Para obtener esta vecindad que contendrá a la región cuadrada con las propiedades mencionadas, basta con generar cuatro vecindades del arreglo cuasiperiódico alrededor de los siguientes puntos:

$$\mathbf{C}_1 = \mathbf{P} + \frac{\sqrt{2}}{2}(1, 1)\, R_S, \quad \mathbf{C}_2 = \mathbf{P} + \frac{\sqrt{2}}{2}(-1, 1)\, R_S,$$

$$\mathbf{C}_3 = \mathbf{P} + \frac{\sqrt{2}}{2}(-1, -1)\, R_S, \quad \mathbf{C}_4 = \mathbf{P} + \frac{\sqrt{2}}{2}(1, -1)\, R_S,$$

donde $\mathbf{P} \in \mathbb{R}^2$ es el punto donde queremos centrar a la región cuadrada. Los vértices correspondientes a la región cuadrada están dados por las siguientes expresiones:

$$\mathbf{V}_1 = \mathbf{P} + \sqrt{2}(1, 1)\, R_S, \quad \mathbf{V}_2 = \mathbf{P} + \sqrt{2}(-1, 1)\, R_S,$$

$$\mathbf{V}_3 = \mathbf{P} + \sqrt{2}(-1, -1)\, R_S, \quad \mathbf{V}_4 = \mathbf{P} + \sqrt{2}(1, -1)\, R_S.$$

Véase la figura 5.5.

Finalmente eliminamos los vértices de los polígonos del arreglo cuasiperiódico que se

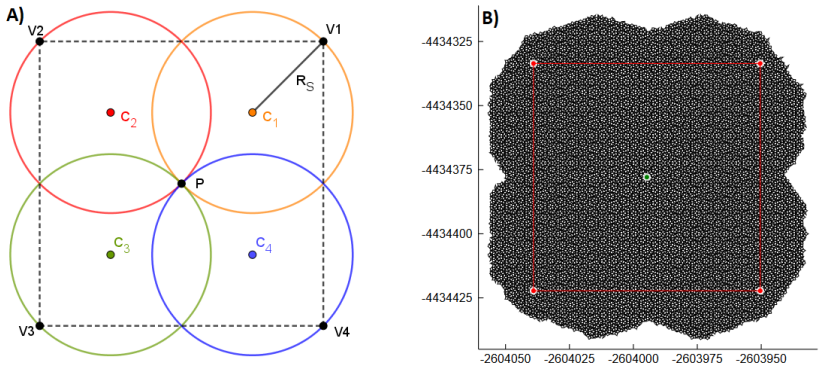

Figura 5.5: A) Idealización de las cuatro vecindades (de radio R_S) generadas alrededor de los puntos $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ y $\mathbf{C}_4$. La región cuadrada se muestra en líneas punteadas y el punto P es el centro de dicha región. B) Vecindad de un arreglo cuasiperiódico de simetría pentagonal generada alrededor del punto $\mathbf{P} = (-2603994.7229115467, -4434377.9162561203)$ con $\alpha = 0.2, \beta = 20, I_V = 20, R_A = 39.0844696836, \gamma = 0.8$; en puntos rojos se muestran los vértices $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ y $\mathbf{V}_4$ de la región cuadrada.

repitan debido a la intersección entre las vecindades generadas alrededor de los puntos $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ y $\mathbf{C}_4$.

Lo expuesto en los párrafos anteriores es, a grandes rasgos, el algoritmo empleado para generar vecindades "cuadradas" de los arreglos cuasiperiódicos de interés. A continuación procedemos a enlistar y describir las diversas funciones que conforman a dicho algoritmo. Una representación visual de la jerarquía de esta parte del código se encuentra en la figura 5.6, en color amarillo aparecen mostradas las funciones o secciones del algoritmo cuya autoría corresponde a Enrique Gómez Cruz, programador del grupo de trabajo del Dr. Atahualpa Kraemer.

- **parche_Cuadrado**: Función encargada de coordinar los diferentes pasos del algoritmo que genera, como producto final, la región cuadrada dentro de una vecindad del arreglo cuasiperiódico de interés centrada en un punto $\mathbf{P} \in \mathbb{R}^2$ dado por el usuario.

  Para ello se definen las variables asociadas a los puntos $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ y $\mathbf{C}_4$; una vez realizado esto se generan, a través de la función **region_Local_Voronoi**, un conjunto de polígonos del arreglo cuasiperiódico tomando como punto central a cada uno de los puntos $\mathbf{C}_i$.

  Tras generar cada uno de estos conjuntos de polígonos se calcula, empleando la función **centroides**, el conjunto de centroides de dichos polígonos, así como un diccionario que relacione cada centroide con los vértices de su polígono asociado.

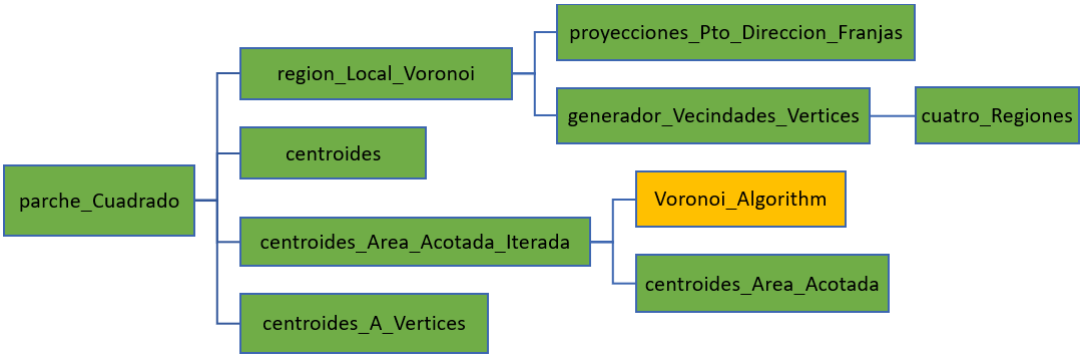

Figura 5.6: Jerarquía del algoritmo para generar vecindades "cuadradas" de algún arreglo cuasiperiódico.

A partir de cada conjunto de centroides se obtiene, utilizando el algoritmo basado en el área de las celdas de Voronoi implementado en la función **centroides_Area_Acotada_Iterada**, la vecindad del arreglo cuasiperiódico con centro en su respectivo punto $\mathbf{C}_i$.

Tras lo anterior se unen los cuatro conjuntos de centroides asociados a los polígonos del clúster central y se eliminan aquellos centroides repetidos. Finalmente con ayuda de los diccionarios creados y la función **centroides_A_Vertices** se recuperan las coordenadas de los vértices de los polígonos correspondientes a los centroides. Dichas coordenadas de los vértices se guardan en un arreglo y se regresan al usuario.

- **region_Local_Voronoi**: Función encargada de, dado un punto $\mathbf{P} \in \mathbb{R}^2$ y los parámetros que definen a un arreglo cuasiperiódico (su simetría rotacional, su conjunto de vectores estrella, etc.), generar una vecindad del arreglo cuasiperiódico centrada en dicho punto.

  Para ello la función obtiene una aproximación, a través de la función **proyecciones_Pto_Direccion_Franjas**, de los números enteros n_i asociados a los vectores estrella $\mathbf{e}_i$ tal que, al sustituir estos números enteros en las expresiones 3.6 - 3.9 para los vértices de los polígonos del arreglo cuasiperiódico, nos generan uno de los polígonos candidatos a ser el polígono contenedor del punto $\mathbf{P}$.

  Empleando la función **generador_Vecindades_Vertices**, se itera la expresión analítica para los vértices de los polígonos del arreglo cuasiperiódico con las diferentes combinaciones de duplas de números enteros (m_i, m_j) correspondientes a las duplas de vectores estrella $(\mathbf{e}_i, \mathbf{e}_j)$, donde el número entero m_i pertenece al

conjunto $\{n_i - \beta, n_i - (\beta - 1), \ldots, n_i - 1, n_i, n_i + 1, \ldots, n_i + \beta - 1, n_i + \beta\}$ con β un número entero asociado al margen de error que permitimos para los números enteros n_i.

Las coordenadas de los vértices de los polígonos del arreglo cuasiperiódico obtenidas en el párrafo anterior se guardan en un arreglo y se regresan al usuario.

- **proyecciones_Pto_Direccion_Franjas**: Función que regresa un arreglo de números reales que aproximan a los números enteros n_i asociados a los vectores estrella $\mathbf{e}_i$ que aparecen en las expresiones 3.6 - 3.9 para los vértices de los polígonos del arreglo cuasiperiódico. Los polígonos generados a partir de sustituir los números enteros n_i en dicha expresión analítica son los candidatos a contener el punto $\mathbf{P}$ dado por el usuario.

  Para ello la función itera sobre los vectores estrella que definen al arreglo cuasiperiódico y proyecta cada uno de estos vectores estrella $\mathbf{e}_i$ con el punto $\mathbf{P}$, el resultado de dicha proyección es divido por la separación entre las "franjas ortogonales" (estudiadas en el capítulo 4) al vector estrella en cuestión. Para el caso de los arreglos cuasiperiódicos generados a partir de vectores estrella que apuntan a los vértices de un polígono regular de N lados, dicha separación entre las franjas es $N/2$.

  Los números reales resultado de estas operaciones se guardan en un arreglo y se regresan al usuario.

- **generador_Vecindades_Vertices**: Función que toma al conjunto de números reales que aproximan a los números enteros n_i y, a partir de los parámetros que definen al arreglo cuasiperiódico así como un margen de error β, regresa los vértices de un conjunto de polígonos del arreglo cuasiperiódico.

  La función itera sobre las diferentes parejas de vectores estrella no colineales $(\mathbf{e}_i, \mathbf{e}_j)$ y, para cada una de estas parejas, itera sobre las diferentes duplas de números enteros (m_i, m_j) con m_i elemento del conjunto $\{n_i - \beta, n_i - (\beta - 1), \ldots, n_i - 1, n_i, n_i + 1, \ldots, n_i + \beta - 1, n_i + \beta\}$. El número entero n_i se obtiene de redondear al entero más cercano la aproximación obtenida por la función **proyecciones_Pto_Direccion_Franjas**.

  Dentro de cada una de las iteraciones mencionadas en el párrafo anterior obtenemos, empleando la función **cuatro_Regiones**, los cuatro vértices que definen a uno de los polígonos del arreglo cuasiperiódico. Estos vértices se almacenan en un arreglo y dicho arreglo se regresa al usuario.

- **cuatro_Regiones**: Implementación directa de las expresiones 3.6 - 3.9 para obtener los vértices de un polígono dentro del arreglo cuasiperiódico.
- **centroides**: Función encargada de obtener los centroides de un conjunto de polígonos del arreglo cuasiperiódico dados sus vértices; de igual forma la función nos regresa un diccionario que relaciona los centroides con los vértices de su polígono asociado.

  Para ello se itera sobre los polígonos dados y en cada iteración se obtiene su respectivo centroide, dado que los polígonos que conforman a los arreglos cuasiperiódicos son paralelogramos, su centroide se localiza en la intersección de sus diagonales, con lo cual cada una de las coordenadas del centroide está dada por el promedio de las respectivas coordenadas de los vértices del polígono.

  Una vez obtenido el centroide de cada polígono se relaciona, vía un diccionario, sus coordenadas a las coordenadas de los cuatro vértices del polígono asociado al centroide en cuestión y se agregan las coordenadas del centroide a un arreglo.

  Tras finalizar la iteración sobre los polígonos dados, el arreglo con las coordenadas de los centroides y el diccionario generado son regresados al usuario.
- **centroides_Area_Acotada_Iterada**: Función que, dado el conjunto de centroides de los polígonos del arreglo cuasiperiódico, itera un número I_V de veces el algoritmo que elimina los polígonos aislados así como los de las fronteras de los clústers y nos regresa el conjunto de centroides de los polígonos que quedan tras dichas iteraciones.

  En cada una de estas I_V iteraciones se genera, empleando la función **getVoronoiDiagram**, el teselado de Voronoi correspondiente a tomar a los centroides como centros de Voronoi; una vez realizado lo anterior se eliminan, a través de la función **centroides_Area_Acotada**, los centroides cuyas celdas de Voronoi posean un área mayor a un valor A definido por el usuario.

  Una vez finalizadas las iteraciones, se regresa al usuario el conjunto de centroides restante.
- **getVoronoiDiagram**: Esta función forma parte de un conjunto de funciones y algoritmos programados por Enrique Gómez Cruz. A grandes rasgos la función toma como entrada un conjunto de puntos y nos regresa una estructura de datos con la información sobre las celdas de Voronoi generadas a partir del conjunto de puntos dado. Dentro de la información que genera la función se encuentra el área de las celdas de Voronoi, los vértices que conforman a cada celda, el punto del conjunto

dado que generó a dicha celda, así como sus celdas vecinas. El código se encuentra disponible en la página web https://github.com/couscouscricket/VoronoiDiagram.

- **centroides_Area_Acotada**: Implementación del algoritmo para eliminar los polígonos aislados así como los de las fronteras de los clústers de un conjunto de polígonos del arreglo cuasiperiódico.

  Para ello la función itera sobre las celdas de un teselado de Voronoi dado y evalúa si el área correspondiente a la celda en cuestión es menor a un valor A definido previamente por el usuario, si no es el caso el algoritmo pasa a la siguiente celda, si la condición se cumple, la función guarda las coordenadas del centro de Voronoi de la celda en un arreglo y pasa a la siguiente celda.

  Una vez finalizada la iteración, la función nos regresa el arreglo con las coordenadas de los centros de Voronoi.

- **centroides_A_Vertices**: Función que, dado un conjunto de centroides de los polígonos del arreglo cuasiperiódico, nos regresa (empleando los diccionarios generados por la función **centroides**) los vértices de los polígonos asociados a dichos centroides.

5.1.2. Desplazamiento cuadrático medio y distribución de vuelos libres

Una vez desarrollado el código que genera estas regiones cuadradas dentro de una vecindad de los arreglos cuasiperiódicos alrededor de un punto arbitrario $\mathbf{P} \in \mathbb{R}^2$ (de ahora en adelante parches cuadrados), podemos ir generando a voluntad las secciones de estos arreglos conforme sean requeridas, de tal forma que sin importar en qué región del espacio nos encontremos ahora somos capaces de estudiar localmente estos sistemas con una complejidad computacional $\sim O(M^2)$, siendo M la simetría rotacional del arreglo cuasiperiódico.

Una de las diversas aplicaciones a lo descrito en el párrafo anterior consiste en un algoritmo para calcular el desplazamiento cuadrático medio de un arreglo cuasiperiódico con simetría rotacional dada por un polígono regular de N lados, considerando un potencial de interacción de amarre fuerte entre una partícula y los sitios del arreglo cuasiperiódico.

Básicamente estamos interesados en conocer qué tan lejos, respecto a su posición inicial, se encuentra una partícula que se desplaza a través de un arreglo cuasiperiódico como función del tiempo, es por ello que la idea detrás del algoritmo consiste en lo siguiente: Generamos un parche cuadrado centrado en la posición inicial de la partícula y calculamos la trayectoria de la misma a partir del potencial de interacción definido

por el usuario, cada cierto tiempo de vuelo Δt (definido como la distancia recorrida por la partícula entre la velocidad de la misma) calculamos el desplazamiento cuadrático de dicha partícula y lo almacenamos en un arreglo. Si el tiempo de vuelo total T es suficientemente grande, la partícula eventualmente alcanzará la frontera de la región cuadrada en donde se encuentra contenida, cuando eso ocurra generamos un nuevo parche cuadrado centrado en la última posición de la partícula dentro de la región cuadrada previa y continuamos calculando su trayectoria. Finalmente, dado que lo que queremos obtener es el desplazamiento cuadrático medio del sistema, iteramos este algoritmo para diversas velocidades iniciales de la partícula, manteniendo la misma posición inicial, así como para diversas posiciones iniciales.

Como caso particular mostraremos los resultados de aplicar este algoritmo para un potencial de esferas duras, en cuyo caso resulta igualmente interesante conocer la distribución de vuelos libres de una partícula, esto se consigue realizando ligeras modificaciones a las funciones que conforman el algoritmo para calcular el desplazamiento cuadrático medio, estas modificaciones están orientadas principalmente a conocer la posición donde la partícula colisiona con un obstáculo circular centrado en alguno de los sitios del arreglo cuasiperiódico, de tal forma que podamos obtener la distancia que hay entre cada dos colisiones consecutivas y guardar dicha información en un arreglo.

A continuación esbozamos a grandes rasgos el algoritmo central de ambas aplicaciones:

1. Se fijan los parámetros del arreglo cuasiperiódico que queremos estudiar, esto es el número de lados N del polígono regular que determina la simetría rotacional del sistema, la constante de desplazamiento α y el margen de error β.

2. Se generan los vectores estrella $\mathbf{e}_i$ dados por:

$$\mathbf{e}_i = \left(\cos\left(\frac{2\,(i-1)\,\pi}{N} \right), \sin\left(\frac{2\,(i-1)\,\pi}{N} \right) \right),$$

   para $i \in \{1, 2, \ldots, N\}$.

3. Se define el número de posiciones iniciales a considerar, así como el número de velocidades iniciales que se evaluarán en cada una de dichas posiciones. De igual forma se fija el tiempo de vuelo total T de la partícula, un intervalo de tiempo Δt tras el cual se calculará el desplazamiento cuadrático de la partícula y un tiempo de vuelo parcial $t = \Delta t$. Así mismo se genera el arreglo que contendrá el desplazamiento cuadrático medio de nuestro sistema como función del tiempo de vuelo.

4. Se genera una posición inicial válida para nuestra partícula considerando el potencial de interacción que el usuario defina (el algoritmo calcula una posición

aleatoria con una distribución uniforme dentro de un cuadrado centrado en el origen de lado L, el usuario debe proporcionar una función que, dado un punto en el plano y los vértices del polígono del arreglo cuasiperiódico que contiene a dicho punto, determine si el punto es una posición inicial válida para la partícula).

5. Dada una posición inicial válida se genera un parche cuadrado centrado en dicha posición y se genera el teselado de Voronoi correspondiente a considerar a los sitios del arreglo cuasiperiódico que conforman a dicho parche como centros de Voronoi. Una vez generado el teselado de Voronoi identificamos la celda de Voronoi que contiene a nuestra partícula en su posición inicial.

6. Se genera una velocidad inicial aleatoria con una distribución uniforme sobre un círculo unitario y calculamos la trayectoria que seguirá la partícula dentro de la celda de Voronoi que la contiene considerando el potencial de interacción entre el sitio del arreglo cuasiperiódico asociado a dicha celda y la partícula. En esta parte el usuario deberá proporcionar la función que determine la posición y la velocidad con la cual la partícula abandona la celda de Voronoi que la contiene, así como el tiempo de vuelo que esto le lleva y los vértices que conforman al lado de la celda por donde la partícula sale; lo anterior a partir de conocer la posición y velocidad de la partícula, los vértices ordenados en sentido horario o antihorario que conforman a la celda de Voronoi contenedora, el lado por donde entró a dicha celda, el sitio del arreglo cuasiperiódico que dio lugar a la celda y un diccionario que relacione los lados de la celda con los sitios del arreglo cuasiperiódico que generan a las celdas vecinas.

7. Si el tiempo de vuelo que le toma a la partícula abandonar la celda que la contiene es menor que el tiempo de vuelto parcial restante t, se acepta el cambio y se actualizan la posición y la velocidad de la partícula, así como la información sobre la celda de Voronoi que contiene a la partícula y se resta el tiempo de vuelo requerido para salir de la celda a la variable t; caso contrario se actualiza la posición y la velocidad de la partícula dentro de la celda contenedora hasta que el tiempo de vuelo parcial restante t se agote. En este punto el usuario debe proporcionar la función que genere la posición y velocidad de la partícula tras un tiempo de vuelo t dado, así como los vértices que conforman al lado de la celda por donde entró la partícula a la celda contenedora actual o, en caso de que haya colisiones con algún obstáculo dentro de la celda, el arreglo $[[Inf, Inf], [Inf, Inf]]$.

8. Cada que el tiempo de vuelo parcial t se vuelve cero, se calcula el desplazamiento cuadrático de la partícula, se promedia con las cantidades correspondientes generadas con otras condiciones iniciales y se vuelve a fijar el tiempo de vuelo restante $t = \Delta t$.

9. Si la celda a la cual la partícula pasaría tras algún tiempo de vuelo no forma parte de alguno de los parches cuadrados generados, se guardan las últimas condiciones de la partícula y se genera un nuevo parche cuadrado centrado en la última

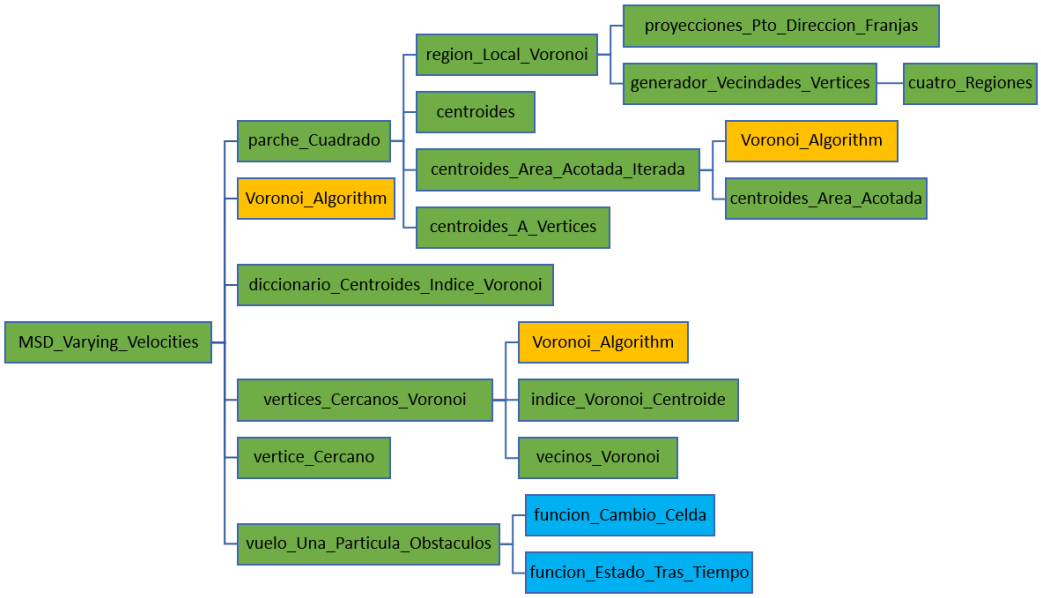

Figura 5.7: Jerarquía del algoritmo para calcular el desplazamiento cuadrático medio de un arreglo cuasiperiódico con un potencial de interacción de amarre fuerte entre una partícula y los sitios del arreglo.

posición de la partícula dentro de la región cuadrada previa. Tras esto se vuelve a generar el teselado de Voronoi de todos los sitios del arreglo cuasiperiódico que se han obtenido a través de los parches cuadrados, se obtiene la celda de Voronoi que contiene a la partícula y se continúa con el cálculo de la trayectoria de la partícula hasta agotar su tiempo de vuelo total T.

10. Se iteran los pasos 6 - 9 el número de velocidades iniciales diferentes que el usuario haya fijado en el paso 3. Al finalizar se promedia el desplazamiento cuadrático medio generado a partir de la presente posición inicial con las respectivas cantidades obtenida a partir de diferentes posiciones iniciales.

11. Por último se itera desde el paso 4 del presente algoritmo un número de veces igual al número de posiciones iniciales diferentes que el usuario fijó en el paso 3.

Una vez planteado el algoritmo para calcular el desplazamiento cuadrático medio procedemos a detallar las funciones que conforman su implementación computacional. Una representación visual de la jerarquía de esta parte del código se encuentra en la figura 5.7, en color amarillo aparecen mostradas las funciones o secciones del algoritmo cuya autoría corresponde a Enrique Gómez Cruz, programador del grupo de trabajo del Dr. Atahualpa Kraemer y en azul aparecen indicadas las funciones que el usuario debe proporcionar.

- **MSD_Varying_Velocities**: Función encargada de calcular el desplazamiento cuadrático medio de un arreglo cuasiperiódico a partir de una posición inicial

fija. A grandes rasgos esta función contiene y coordina el algoritmo desde el paso 5 tal como se redactó en párrafos anteriores.

Para ello genera un parche cuadrado, a través de la función **parche_Cuadrado**, alrededor de un punto $\mathbf{P} \in \mathbb{R}^2$ correspondiente a la posición inicial de la partícula que se desplazará por el arreglo cuasiperiódico.

Se eliminan todos los vértices repetidos de los polígonos del arreglo cuasiperiódico generados por el parche cuadrado y, a partir de ellos, se obtiene un diccionario que nos indique qué vértices están dentro de la región cuadrada definida en la subsección previa.

Una vez generado el diccionario se obtiene el teselado de Voronoi, empleando la función **getVoronoiDiagram**, correspondiente a considerar a los vértices únicos del parche cuadrado como los centros de Voronoi y se genera un diccionario, a través de la función **diccionario_Centroides_Indice_Voronoi** que relacione el sitio del arreglo cuasiperiódico con el índice de la celda de Voronoi asociada a dicho sitio.

Tras lo anterior se define un arreglo en donde se almacenarán los centros de cada uno de los parches cuadrados generados.

A través de la función **vertices_Cercanos_Voronoi** se obtiene un arreglo con las coordenadas de los sitios del arreglo cuasiperiódico más cercanos a la posición inicial y, con la función **vertice_Cercano**, se determina cuál es el sitio más cercano a dicha posición.

Una vez realizado los pasos previos se itera sobre el número de velocidades iniciales diferentes que el usuario haya determinado. A partir de este punto todo lo descrito a continuación corresponde a los pasos realizados en cada iteración salvo que se indique lo contrario.

Se genera una velocidad inicial aleatoria con una distribución uniforme sobre el círculo unitario, se obtiene el índice de la celda de Voronoi que contiene a la partícula y dado que la partícula no llegó a la celda actual procedente de otra celda, las coordenadas de los vértices del lado por el que la partícula entra se definen como $[[Inf, Inf], [Inf, Inf]]$.

Se itera sobre el número de vuelos parciales que conforman el vuelo total de la partícula. Salvo que se indique lo contrario, lo que viene a continuación ocurre

dentro de cada iteración de un vuelo parcial.

Se define el tiempo de vuelo parcial $t = \Delta t$ y se calcula la trayectoria de la partícula a través del arreglo cuasiperiódico empleando la función **vuelo_Una_Particula_Obstaculos**, en este punto es posible que la trayectoria sea una curva cerrada (esto dependerá del tipo de potencial de interacción que defina el usuario) en cuyo caso se termina la iteración sobre los vuelos parciales y se pasa a una siguiente iteración con una velocidad inicial diferente.

Si la trayectoria no resultó ser una curva cerrada, pero la partícula llegó a la frontera de la región cuadrada donde se encuentra contenida sin terminar su tiempo de vuelo parcial, se genera un nuevo parche cuadrado empleando la función **parche_Cuadrado**, se agrega el centro del nuevo parche cuadrado a su respectivo arreglo, se eliminan los sitios del arreglo cuasiperiódico que estén repetidos y con ellos se actualiza el diccionario que determina qué sitios están dentro de alguna región cuadrada.

Empleando la función **getVoronoiDiagram** se obtiene el nuevo teselado de Voronoi del espacio considerando a los sitios del arreglo cuasiperiódico sin repetir como los centros de Voronoi y se actualiza el diccionario, a través de la función **diccionario_Centroides_Indice_Voronoi**, que relaciona el sitio del arreglo cuasiperiódico con el índice de la celda de Voronoi que genera.

Se obtiene el índice de la celda de Voronoi que contiene a nuestra partícula y se continúa calculando el resto de su trayectoria, empleando la función **vuelo_Una_Particula_Obstaculos**, hasta agotar su tiempo de vuelo parcial. Tras agotar su tiempo de vuelo parcial se obtienen las coordenadas del sitio del arreglo cuasiperiódico más cercano a nuestra partícula.

Una vez finalizada la trayectoria parcial de la partícula se calcula el desplazamiento cuadrático de la misma y se almacena dicho valor en un arreglo, promediando dicho valor con los correspondientes valores obtenidos con otras velocidades iniciales diferentes, con lo cual se finaliza una iteración sobre el número de vuelos parciales.

Tras terminar cada trayectoria total de la partícula se añade un uno a un contador para saber cuántas trayectorias no cerradas se realizaron y con ello se finaliza una iteración sobre el número de velocidades iniciales diferentes.

Por último se regresa al usuario el arreglo con el desplazamiento cuadrático medio del sistema y el número de trayectorias no cerradas con el que se promedió el

desplazamiento cuadrático medio.

- **diccionario_Centroides_Indice_Voronoi**: Función que genera un diccionario para relacionar el sitio del arreglo cuasiperiódico con el índice asociado a la celda de Voronoi correspondiente dicho sitio.

  Para ello la función genera un diccionario vacío e itera sobre cada una de las celdas del teselado de Voronoi, obteniendo en cada iteración las coordenadas del sitio que da lugar a la celda en cuestión. Con esta información se añade una entrada al diccionario donde la llave es las coordenadas del sitio del arreglo cuasiperiódico y a dicha llave le asocia el número de la iteración actual. Tras finalizar con las iteraciones se regresa al usuario el diccionario.

- **vertices_Cercanos_Voronoi**: Función que obtiene las coordenadas de los sitios del arreglo cuasiperiódico más cercanos a un punto $\mathbf{P} \in \mathbb{R}^2$ contenido dentro de una vecindad dada de dicho arreglo.

  La función comienza agregando el punto **P** al conjunto de sitios de la vecindad del arreglo cuasiperiódico dado, con este nuevo conjunto de puntos se genera, empleando la función **getVoronoiDiagram**, el teselado de Voronoi correspondiente a tomar como centros de Voronoi a los puntos del conjunto previamente mencionado.

  Usando la función **indice_Voronoi_Centroide** se obtiene el índice de la celda de Voronoi asociada al punto **P** y, con la función **vecinos_Voronoi**, se recuperan las coordenadas de los sitios del arreglo cuasiperiódico que dan lugar a las celdas de Voronoi vecinas a la celda asociada a **P**. Este conjunto de puntos son los sitios del arreglo cuasiperiódico más cercanos al punto **P**.

  Finalmente se elimina al punto **P** del conjunto de sitios del arreglo cuasiperiódico que conforman a la vecindad dada y se regresa al usuario el conjunto de sitios del arreglo cuasiperiódico más cercanos a **P**.

- **indice_Voronoi_Centroide**: Función que, dado un punto $\mathbf{P} \in \mathbb{R}^2$ que forma parte del conjunto de centros de un teselado de Voronoi, nos regresa el índice de la celda de Voronoi correspondiente al punto **P**.

  Para ello la función simplemente itera sobre todas las celdas de Voronoi hasta que localiza aquella cuyo centro de Voronoi asociado sea el punto **P**, momento en el cual regresa al usuario el valor de la iteración en la que se encontró la coincidencia.

- **vecinos_Voronoi**: Función que, dado el índice de una celda dentro de un teselado de Voronoi, nos regresa las coordenadas de los centros de Voronoi de sus celdas

vecinas.

La función comienza generando un arreglo vacío donde guardar las coordenadas de los centros de las celdas vecinas, después itera sobre los lados que conforman a la celda en la que estamos interesados y, en cada iteración, obtiene el centro de Voronoi de la celda que comparte el lado correspondiente a la iteración actual, guardando las coordenadas de dicho centro en el arreglo correspondiente.

Tras finalizar la iteración se regresa al usuario el arreglo con las coordenadas de los centros de Voronoi.

- **vertice_Cercano**: Función que, dado un conjunto de puntos en el plano y un punto $\mathbf{P} \in \mathbb{R}^2$, nos regresa el punto más cercano a $\mathbf{P}$ de dicho conjunto.

  Para ello la función itera sobre todos los puntos que conforman al conjunto dado y calcula en cada iteración la distancia euclidiana que existe entre $\mathbf{P}$ y el punto en cuestión, este valor se guarda en un arreglo.

  Tras iterar todos los puntos se obtiene la mínima de las distancias y se regresa al usuario el elemento del conjunto de puntos correspondiente a dicho valor.

- **vuelo_Una_Particula_Obstaculos**: Función que calcula la trayectoria de una partícula que se desplaza dentro de un arreglo cuasiperiódico en un intervalo de tiempo.

  Dado un tiempo de vuelo $t = \Delta t$, la función iterará los siguientes pasos hasta que se cumpla la condición $t = 0$:

  Se obtiene el sitio del arreglo cuasiperiódico asociado a la celda de Voronoi que contiene a la partícula y se verifica si dicho sitio está dentro de alguna región cuadrada de los parches cuadrados que se han generado; si el sitio no cae dentro de alguna región cuadrada se regresa al usuario la posición y velocidad de la partícula, el tiempo de vuelo restante t, el índice de la celda de Voronoi que contiene a la partícula y los vértices del segmento por donde entró la partícula a la celda actual.

  Si el sitio cae dentro de alguna región cuadrada se itera sobre los lados de la celda de Voronoi que contiene a la partícula y se genera un diccionario que relaciona los vértices de cada lado con el índice de la celda vecina que comparte dicho lado con la celda contenedora, así como un diccionario que relaciona esos mismos vértices con el sitio del arreglo cuasiperiódico asociado a esa misma celda vecina.

Se calcula la trayectoria de la partícula dentro de la celda contenedora a partir de una **funcion_Cambio_Celda** como la descrita en el paso 6 del algoritmo presente en esta sección. Para obtener la distribución de vuelos libres con un potencial de interacción de esferas duras, se añade a la salida de **funcion_Cambio_Celda** la variable que nos indique la posición donde colisiona la partícula dentro de la celda de Voronoi, si no hay colisión se regresa el arreglo $[Inf, Inf]$.

Si el tiempo de vuelo requerido para abandonar la celda actual es infinito, la trayectoria es una trayectoria cerrada y en ese caso se arroja una excepción al código, caso contrario se recuperan las coordenadas de los vértices del lado de la celda por donde la partícula sale y, con ayuda de los diccionarios generados, se obtiene el índice de la celda de Voronoi a la cual la partícula va a entrar.

Si el tiempo de vuelo empleado en salir de la celda contenedora es menor que el tiempo de vuelo restante t, se acepta la trayectoria, se resta al parámetro t el tiempo de vuelo empleado en salir de la celda y se actualiza la posición y velocidad de la partícula, así como el índice de la celda contenedora y los vértices del lado por el cual la partícula ha entrado a su actual celda contenedora. Para conocer la distribución de vuelos libres con un potencial de interacción de esferas duras, si en la actual celda de Voronoi la partícula colisionó, se calcula la distancia entre el punto de colisión dentro de la actual celda y el anterior punto de colisión (si es la primera colisión se considera a la posición inicial como el anterior punto de colisión) y se guarda dicha información en un arreglo.

Si por el contrario el tiempo de vuelo empleado en salir de la celda contenedora es mayor que el tiempo de vuelo restante t, se calcula a través de una **funcion_Estado_Tras_Tiempo** como la descrita en el paso 7 del algoritmo presente en esta sección las condiciones de la partícula tras un tiempo de vuelo t. Para conocer la distribución de vuelos libres con un potencial de interacción de esferas duras, se añade a la salida de **funcion_Estado_Tras_Tiempo** la variable que nos indique la posición donde colisiona la partícula dentro de la celda de Voronoi, si no hay colisión se regresa el arreglo $[Inf, Inf]$ y se realiza el algoritmo descrito al final del párrafo anterior.

Cuando el tiempo de t sea cero, se regresa al usuario la posición y velocidad de la partícula, el tiempo de vuelo $t = 0$, el índice de la celda de Voronoi que contiene a la partícula y los vértices del lado por el cual la partícula entró a dicha celda (o un arreglo $[[Inf, Inf], [Inf, Inf]]$ si después de entrar a la actual celda de Voronoi ha ocurrido una colisión con algún obstáculo).

5.2. Implementación del código

Empleando las funciones y algoritmos definidos en la sección previa estamos en condiciones de simular la trayectoria de una partícula dentro de cualquier arreglo cuasiperiódico con simetría rotacional N una vez proporcionadas las funciones que calculan, dentro de una celda de Voronoi, la trayectoria de la partícula sujeta a un potencial de interacción de amarre fuerte entre dicha partícula y el centro de Voronoi de la celda. Como un caso particular mostraremos en la presente sección una de estas trayectorias considerando un arreglo cuasiperiódico tipo Penrose (simetría pentagonal) y un potencial de interacción de esferas duras.

Los parámetros empleados para simular la trayectoria que se muestra en la figura 5.8 fueron los siguientes:

- Simetría del arreglo cuasiperiódico, $N = 5$.
- Vectores estrella $\mathbf{e}_i = \left(\cos\left(\frac{2(i-1)\pi}{N}\right), \sin\left(\frac{2(i-1)\pi}{N}\right)\right), \forall\ i \in \{1,2,3,4,5\}$.
- Desplazamiento respecto al origen, $\alpha_i = 0.2,\ \forall i \in \{1,2,3,4,5\}$.
- Margen de error, para los números enteros n_i, $\beta_i = 20,\ \forall i \in \{1,2,3,4,5\}$.
- Cota superior $A = 1.2$ para el área de las celdas de Voronoi asociadas a los centroides de los polígonos interiores a un clúster del arreglo cuasiperiódico.
- Número de iteraciones del algoritmo para eliminar los polígonos aislados y de la capa externa de los clústers, $I_V = 20$.
- Radio del círculo que aproxima a las vecindades del arreglo cuasiperiódico, $R_A = 39.184$
- Número de vecindades analizadas para obtener el radio R_A, $I = 100$.
- Parámetro para escalar el radio R_A, $\gamma = 0.8$
- Radio de los obstáculos centrados en los sitios del arreglo cuasiperiódico, $R = 0.05$
- Tiempo de vuelo total de la partícula, $T = 250$.
- Intervalo de tiempo para guardar desplazamiento cuadrático de la partícula, $\Delta t = 10$.
- Posición inicial de la partícula, $\mathbf{P} = (105545.98124870296, -767696.4914500959)$.
- Velocidad inicial de la partícula, $\mathbf{V} = (0.09714817272937593, 0.9952699294841291)$.

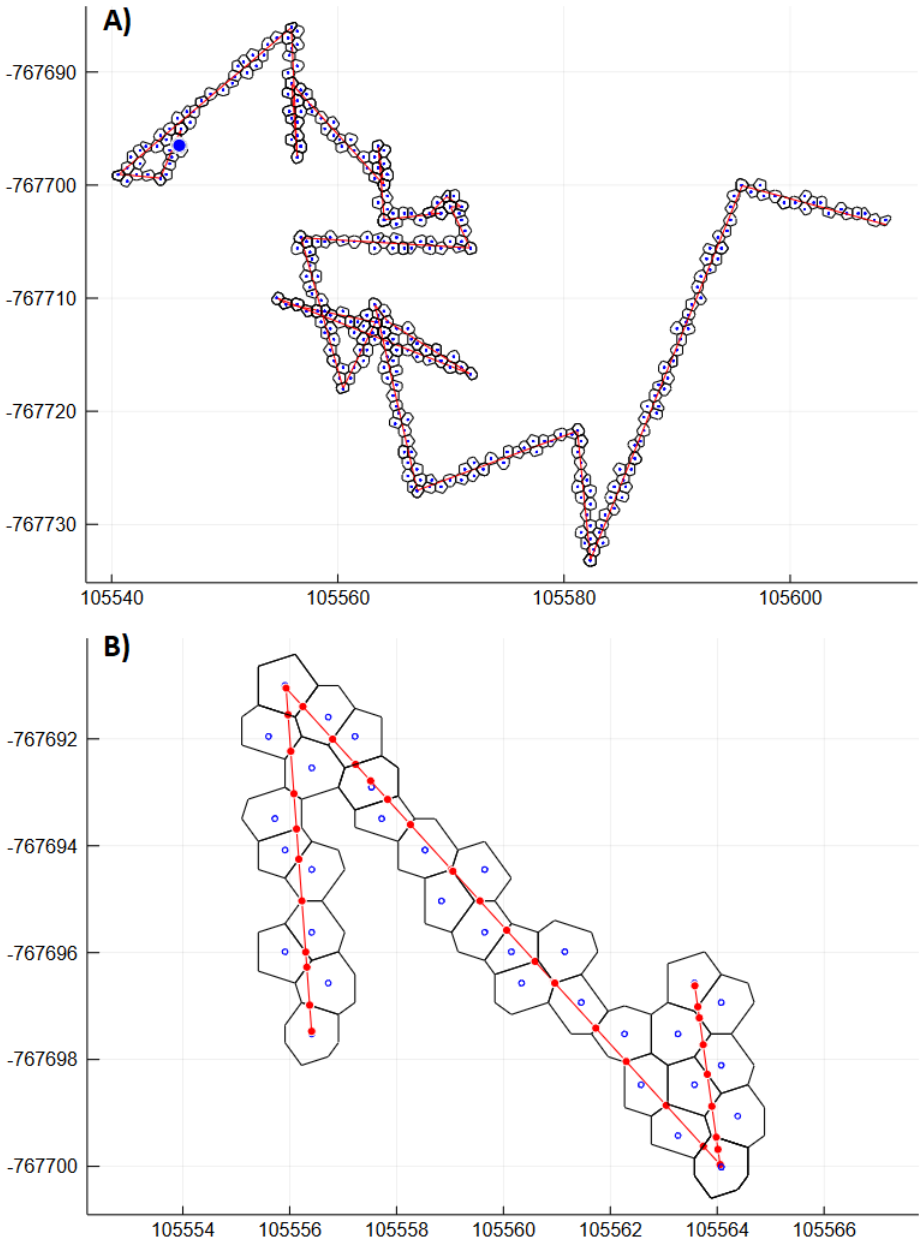

Figura 5.8: A) Trayectoria de una partícula sujeta a un potencial de esferas duras dentro de un arreglo cuasiperiódico con simetría rotacional $N = 5$. El radio de los obstáculos es $r = 0.05$ y el tiempo de vuelo total fue $T = 250$. La posición inicial de la partícula se indica con un gran punto azul, cuyas dimensiones son únicamente para facilitar su visualización, mientras que los pequeños puntos azules son los obstáculos. **B)** Segmento de la trayectoria mostrada en A), los puntos en rojo muestran la posición de la partícula al salir de cada celda de Voronoi, al colisionar con un obstáculo o cuando un intervalo de tiempo Δt ha transcurrido.

Capítulo 6

Resultados numéricos

A lo largo de los capítulos previos hemos desarrollado los aspectos teóricos más relevantes sobre los ambientes cuasiperiódicos y cómo generarlos, centrándonos principalmente en el desarrollo de un código eficiente que nos permita generar vecindades, alrededor de un punto arbitrario en el plano, de un arreglo cuasiperiódico. En el presente capítulo mostraremos algunos resultados obtenidos a partir de la implementación del código expuesto en el capítulo anterior basado en nuestro algoritmo presentado en el capítulo 2.

Cabe mencionar que los resultados que se mostrarán a continuación no representan el producto principal del la presente tesis, siendo este el algoritmo y el código anteriormente mencionados, sino que forman parte de las distintas aplicaciones que pueden realizarse a través del uso de nuestro trabajo aquí expuesto.

6.1. Eficiencia computacional

Una de las principales ventajas del algoritmo computacional que hemos desarrollado es su eficiencia para obtener los puntos que conforman una vecindad arbitraria de un arreglo cuasiperiódico. Como muestra de lo anterior, hemos implementado la siguiente prueba:

Sea P un punto arbitrario en el plano tal que su distancia al origen es d y sea $t(d)$ el tiempo que tarda una computadora en encontrar el polígono de un arreglo cuasiperiódico dado, tal que este contenga al punto P; véase la figura 6.1. La prueba consiste en generar la gráfica de $t(d)$ al ir variando los valores de d implementando nuestro algoritmo para generar vecindades de un arreglo cuasiperiódico (en lo posterior, algoritmo descentralizado; véase la sección 5.1.1. de la presente tesis), y compararla con la gráfica generada al utilizar un algoritmo ingenuo, el cual consiste en aplicar directamente las expresiones 3.6 - 3.9, obtenidas durante el capítulo 3, para generar vecindades del arreglo cuasiperiódico centradas en el origen.

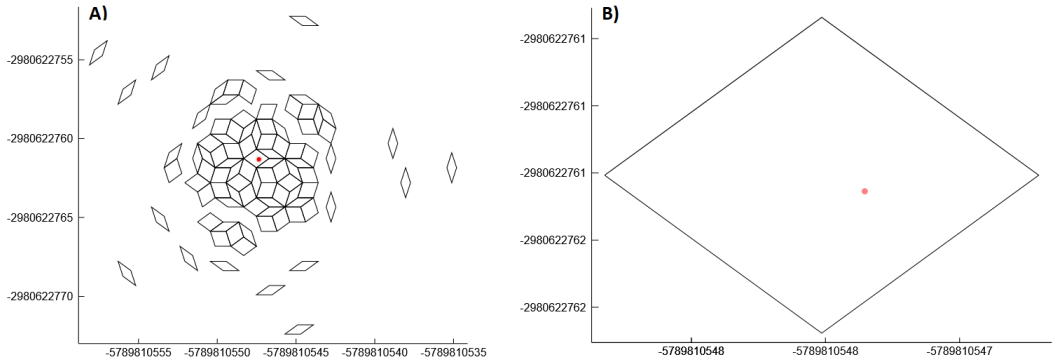

Figura 6.1: A) Vecindad de un arreglo cuasiperiódico tipo Penrose alrededor de un punto rojo. **B)** Polígono del arreglo cuasiperiódico tipo Penrose que contiene al punto rojo.

El algoritmo empleado para encontrar el polígono del arreglo cuasiperiódico que contiene al punto P consiste en lo siguiente:

1. Sea $\mathbb{A}$ el conjunto de todos los polígonos de la vecindad del arreglo cuasiperiódico de interés. Sea SR una semirecta con dirección arbitraria que parte del punto P.

2. Sea $x \in \mathbb{A}$ un polígono arbitrario. Si el número de intersecciones de SR con las aristas de x es impar, entonces el punto P está contenido por x y el algoritmo termina, regresando el polígono contenedor. Si el número de intersecciones de SR con las aristas de x es par (o cero), el punto P no está contenido por x y se itera este paso seleccionando otro polígono del conjunto $\mathbb{A}$. Véase la figura 6.2.

En la figura 6.3 se muestran las gráficas del tiempo de cómputo $t(d)$ como función de la distancia d del punto P en un arreglo cuasiperiódico tipo Penrose empleando el algoritmo ingenuo (imagen A) y empleando el algoritmo descentralizado (imagen B). Debido al carácter aleatorio del algoritmo para encontrar el polígono que contiene al punto P, cada punto de las gráficas fue obtenido a partir de promediar los tiempos obtenidos en diferentes iteraciones, siendo 20 iteraciones para el caso de la imagen A y 30 iteraciones para el caso de la imagen B. La prueba fue realizada en una laptop con procesador Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz y 23.8 GB de RAM.

De las gráficas contenidas en 6.3 podemos observar cómo el tiempo de cómputo $t(d)$ empleando el algoritmo ingenuo crece siguiendo una ley de potencias cuyo exponente es 2.377, mientras que empleando nuestro algoritmo el tiempo de cómputo se mantuvo prácticamente constante (considerando el ruido generado por los diversos procesos de fondo que lleva a cabo la computadora) con valores entre 0.01 y 0.0025 segundos.

Otro aspecto relevante a considerar está en los valores empleados para el parámetro d en ambos casos. Para el caso de la figura A el parámetro d fue tal que $d \in \{1.1^n \mid n \in$

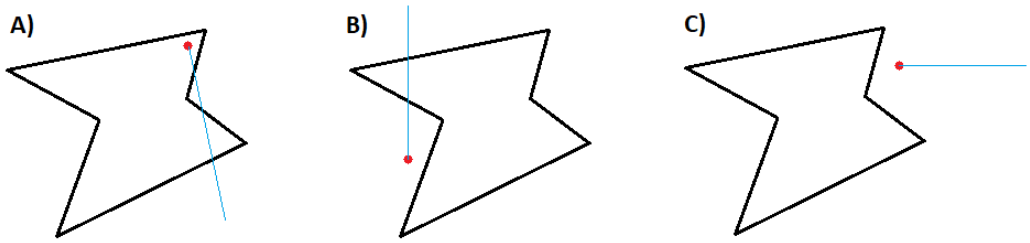

Figura 6.2: Ejemplo del algoritmo para determinar si un punto está dentro o fuera de un polígono dado. En la imagen A) el punto se encuentra dentro del polígono y una semirecta que parte de dicho punto intersecta con tres aristas del polígono. En el caso B) el punto se encuentra fuera y una semirecta que parte de dicho punto intersecta con dos aristas del polígono. En el caso C) el punto se encuentra fuera y una semirecta que parte de dicho punto no intersecta arista alguna del polígono.

$\{1, 2, \ldots, 70\}\}$ mientras que para el caso de la figura B se consideraron los valores $d \in \{1.1^n \mid n \in \{1, 2, \ldots, 200\}\}$. Esta diferencia en el conjunto de valores empleados obedece a cuestiones relacionadas con la RAM de la computadora debido a que, para el caso con $d = 1.1^{70} \approx 790$ la vecindad del arreglo cuasiperiódico generado aplicando el algoritmo ingenuo requiere un aproximado de 10 GB de RAM, mientras que dicha limitante no aparece al aplicar nuestro algoritmo, siendo la memoria empleada para cualquier valor de d del orden de los 500 kB.

6.2. Difusión en ambientes cuasiperiódicos

El principal resultado de la sección anterior nos indica que, haciendo uso del algoritmo desarrollado en el capítulo previo, el tiempo de cómputo para generar una vecindad de un arreglo cuasiperiódico alrededor de un punto P es independiente de la distancia de dicho punto al origen; este resultado es el que nos permite estudiar de manera directa la difusión en sistemas cuasiperiódicos, ya que no estamos limitados a permanecer en una vecindad de dicho sistema alrededor del origen, sino que podemos ir generando diferentes vecindades del sistema en cuestión conforme nos vamos desplazando por este, sin penalización en los tiempos de cómputo para generarlas.

Un ejemplo sobre cómo implementar nuestro algoritmo para estos fines fue presentado en la sección 5.2. de la presente tesis, en donde se muestra una trayectoria de una partícula puntual dentro de un arreglo cuasiperiódico tipo Penrose con un potencial de interacción de esferas duras entre la partícula y unos discos de radio $r = 0.05$ unidades centrados en los vértices de los polígonos que conforman al arreglo cuasiperiódico (los detalles sobre los parámetros empleados para generar estas trayectorias se encuentran en esa misma sección).

En la figura 6.4 se muestran los resultados sobre la difusión presente en el sistema

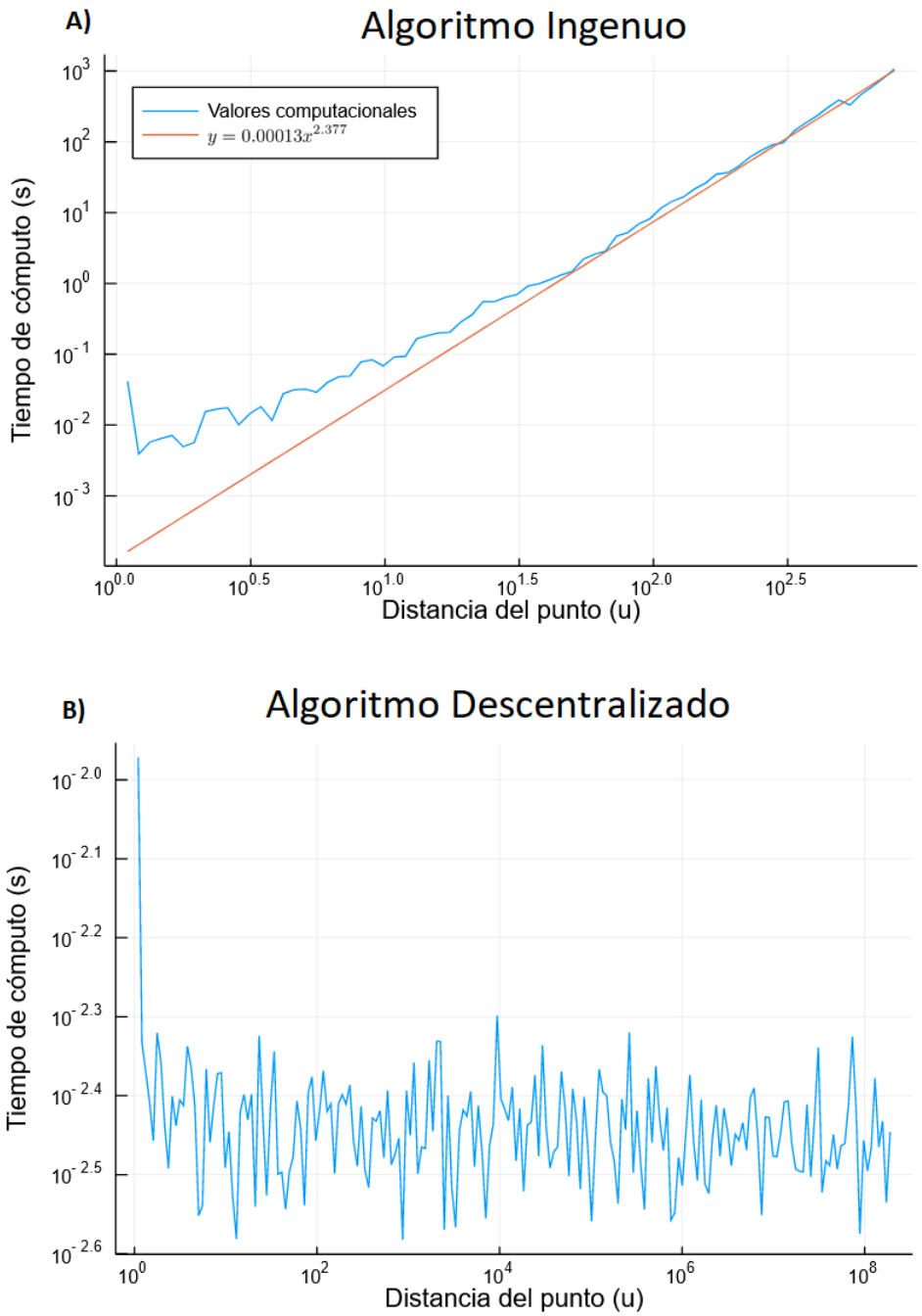

Figura 6.3: Comparativa del tiempo de cómputo empleado para encontrar el polígono de un arreglo cuasiperiódico tipo Penrose que contiene a un punto P a una distancia d del origen. En la imagen A se muestra la gráfica obtenida empleando un algoritmo ingenuo donde cada punto de la gráfica corresponde a un promedio sobre 20 iteraciones. En la imagen B se muestra la gráfica obtenida empleando nuestro algoritmo descentralizado donde cada punto de la gráfica corresponde a un promedio sobre 30 iteraciones.

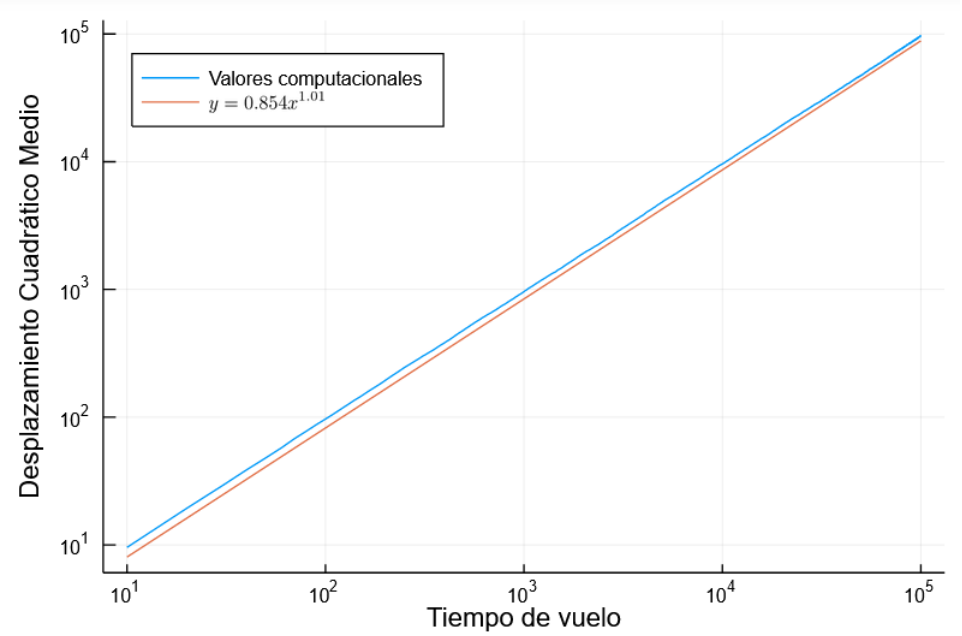

Figura 6.4: Desplazamiento cuadrático medio como función del tiempo de vuelo en escala logarítmica para una partícula en un arreglo cuasiperiódico tipo Penrose de discos duros con radio $r = 0.05$ unidades. Cada punto de la gráfica correspondiente a los valores computacionales se obtuvo promediando 10,000 partículas. La recta $y = 0.854x^{1.01}$ se ha desplazado ligeramente hacia abajo para facilitar su visualización.

descrito en el párrafo anterior, para ello se consideraron 10,000 partículas distribuidas de la siguiente manera:

1. Se consideraron 25 posiciones iniciales distintas, cada una de ellas generadas de manera aleatoria con una distribución equiprobable dentro de un cuadrado de lado $1,000,000$ centrado en el origen.

2. Para cada posición inicial, se consideraron 400 velocidad iniciales distintas, con una distribución equiprobable sobre el círculo unitario.

Cada una de estas trayectorias tuvo como parámetro de tiempo de vuelo total $T = 100,000$ manteniendo el resto de parámetros idénticos a los expuestos en la sección 5.2 del capítulo anterior.

Como se aprecia en la gráfica contenida en 6.4, los resultados obtenidos muestran que el desplazamiento cuadrático medio de una partícula en un arreglo cuasiperiódico de simetría pentagonal bajo un potencial de esferas duras en el cual el radio de los obstáculos es $r = 0.05$ unidades, sigue una ley de potencias cuyo exponente es 1.01; este resultado nos habla de una difusión normal presente en el sistema el cual, según los

resultados reportados en el 2013 por Atahualpa Kraemer y David Sanders [35], presenta un horizonte finito. Sin embargo, el tamaño de los obstáculos es bastante pequeño, lo que hace sospechar que en el caso del gas de Lorentz con un arreglo de simetría pentagonal no se forman canales. Esto podría parecer contradictorio a los resultados de Kraemer y Sanders, sin embargo en dimensiones más altas, el subespacio ortogonal al cilindro (pensando en un método de corte y proyección) puede ser totalmente irracional aunque el cilindro conecte dos de las esquinas opuestas del hipercubo. El arreglo cuasiperiódico de Penrose se forma precisamente de esta forma y, por lo tanto, un cilindro con radio $r \to 0$ intersectará todas las caras del hipercubo, destruyendo los posibles canales.

Por otro lado, en el 2015, A. S. Kraemer, M. Schmiedeberg y D. P. Sanders [47] demostraron que normalmente la distribución de vuelos libres en presencia de canales sifuen una ley de potencias con exponente -3. Para el radio de los cilindros donde los canales se destruyen, esa distribución de vuelos libres pasa a tener un exponente -5, formando un sistema de horizonte localmente finito. Esto fue comprobado numéricamente para un cuasicristal obtenido a partir de proyectar un sistema 3D en uno 2D. Cabe destacar que los argumentos dados por los autores fueron siempre bajo la consideración de que el cilindro tenía un radio mayor a cero. No resulta evidente qué sucede en el caso de sistemas con simetría rotaciona, es decir, donde el cilindro conecta dos de las esquinas opuestas del hipercubo, en el caso donde el radio del cilindro tiende a cero, límite que se conoce como Boltzmann-Grads [48].

6.3. Función de distribución radial

Diferentes estados de la materia poseen propiedades físicas diferentes, producto de la distribución de sus elementos y la interacción entre ellos. Por ejemplo, una manera de distinguir entre sólidos y líquidos es a partir del concepto de hiperuniformidad, el cual se relaciona con las fluctuaciones en la densidad de los sistemas de muchas partículas, más en concreto con la supresión anómala de estas a escalas de longitud grandes. Es un resultado conocido el que los cristales, cuasicristales y algunos sistemas desordenados son hiperuniformes [50].

Otra de las herramientas con las cuales podemos realizar un análisis sobre la distribución de los elementos que conforman a un sistema en particular y determinar si corresponde a un sistema ordenado o a uno desordenado, es la función de distribución radial $g(r)$. De manera general, la función de distribución radial nos brinda información sobre cómo están distribuidos los elementos de un conjunto de partículas (átomos) como función de la distancia, siendo el origen un elemento de dicho conjunto; en otras palabras, la función $g(r)$ nos proporciona una medida para la probabilidad de que se encuentre una partícula a una distancia r de otra [49].

En un cristal, la función $g(r)$ está conformada por picos muy definidos a distancias

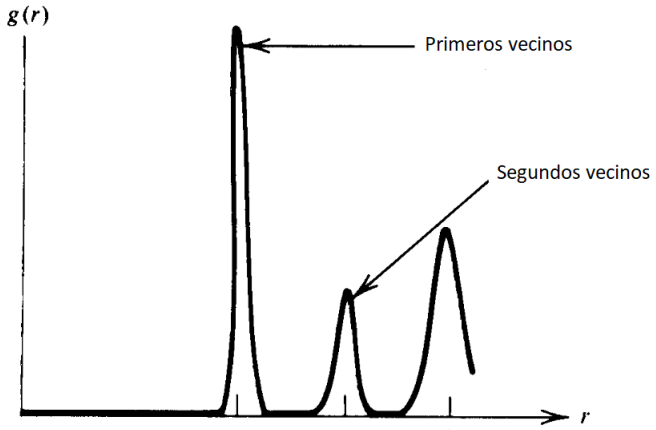

Figura 6.5: Función de distribución radial para un cristal clásico. Imagen obtenida de [49].

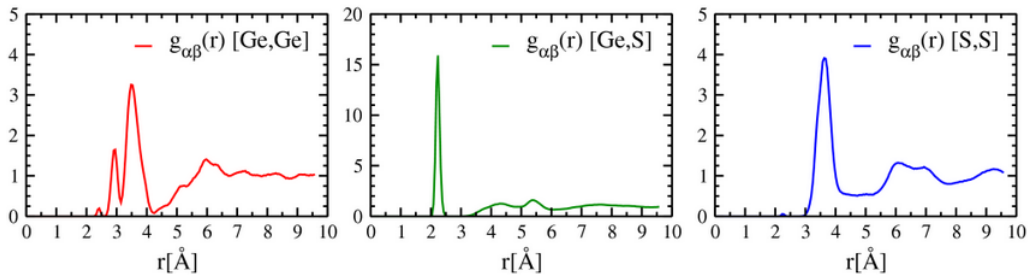

Figura 6.6: Funciones de distribución radial para un vidrio de GeS_2 a una temperatura de 300 K. Imagen recuperada de http://people.cst.cmich.edu/petko1vg/isaacs/phys/rdfs.html

muy especificas (correspondientes a los primeros vecinos, segundo vecinos, etc.) mientras que dicha probabilidad cae a cero rápidamente para cualesquiera otras distancias; véase la figura 6.5. En un vidrio, la probabilidad de encontrar un átomo a una distancia igual al diámetro de estos, es muy alta dado que se encuentran en contacto unos con otros, sin embargo dicha probabilidad decae rápidamente conforme la distancia va en aumento hasta una probabilidad homogénea, similar a la de un gas; véase por ejemplo la figura 6.6.

Relacionado a lo descrito en párrafos anterior y como muestra del poder de cómputo de nuestro algoritmo, hemos obtenido la función de distribución radial $g(r)$ para diversos arreglos cuasiperiódicos al ir variando su simetría rotacional.

Para nuestro caso particular de estudio, el algoritmo empleado para calcular la función $g(r)$ consistió en lo siguiente: Dada una distribución espacial de puntos en el plano, se calculó el promedio de puntos que caen dentro de un anillo de radio inte-

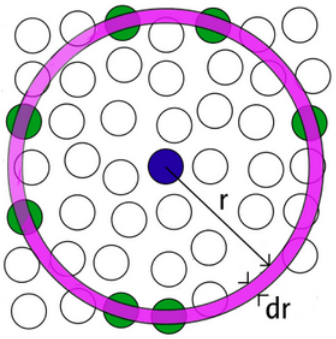

Figura 6.7: Discretización espacial para la evaluación de la función de distribución radial; los discos en verde corresponden a los átomos cuyo centro se encuentra contenido en un anillo de grosor dr y radio interior r. Imagen recuperada de http://people.cst.cmich.edu/petko1vg/isaacs/phys/rdfs.html

rior r y grosor $dr = 0.1$ unidades, tomando como centro del anillo a un elemento del conjunto de puntos dado (véase la figura 6.7); una vez obtenido este promedio, dicho valor fue dividido entre el perímetro del anillo considerando el radio r. El resultado de esta división corresponde al valor de la función de distribución radial para un radio r.

En la imagen 6.8 se muestran las gráficas con las diferentes funciones $g(r)$ para los arreglos cuasiperiódicos de simetría rotacional $N \in \{5, 29, 37, 41, 47, 53, 59, 67, 250\}$. Cada una de estas gráficas fue obtenida a partir de analizar cinco vecindades diferentes del arreglo cuasiperiódico en cuestión, cada una de ellas centradas en algún punto $P = (x, y)$ donde $x, y \in \left[-1\text{x}10^{-6}, 1\text{x}10^{6}\right]$. Una vez generadas las vecindades del arreglo cuasiperiódico, cuidando que en ellas hubiera por lo menos 1500 puntos sobre los cuales aplicar el algoritmo descrito en el párrafo anterior, se agregó un ruido gaussiano con desviación estandar 0.05 a cada uno de los puntos que las conforman.

Como se puede apreciar en la imagen, las funciones $g(r)$ para los arreglos cuasiperiódicos con simetría rotacional mayor o igual a 29 se mantienen prácticamente iguales, destacando en ellas la presencia de un pico muy pronunciado cerca del valor $r = 1$ u y otro pico menor (con un ancho más grande con tendencia al lado izquierdo) cerca del valor $r = 2$ u. Nótese de igual forma como la función decae a un valor ligeramente mayor a 0.1 conforme el radio r aumenta. Esto nos indica una fuerte correlación presente a corto alcance que decae prácticamente a cero (considerando el grosor del anillo $dr = 0.1$) a largo alcance.

Por otro lado, se observa que la función $g(r)$ para el caso de simetría pentagonal tiene bastantes picos muy bien definidos, los cuales no parecen desaparecer conforme el parámetro r aumenta. En conjunto, estos resultan nos indican que a simetrías bajas los arreglos cuasiperiódicos tienen una $g(r)$ similar a la de los cristales, mientras que para simetrías altas, la función $g(r)$ se muestra similar a la de un sistema desordenado.

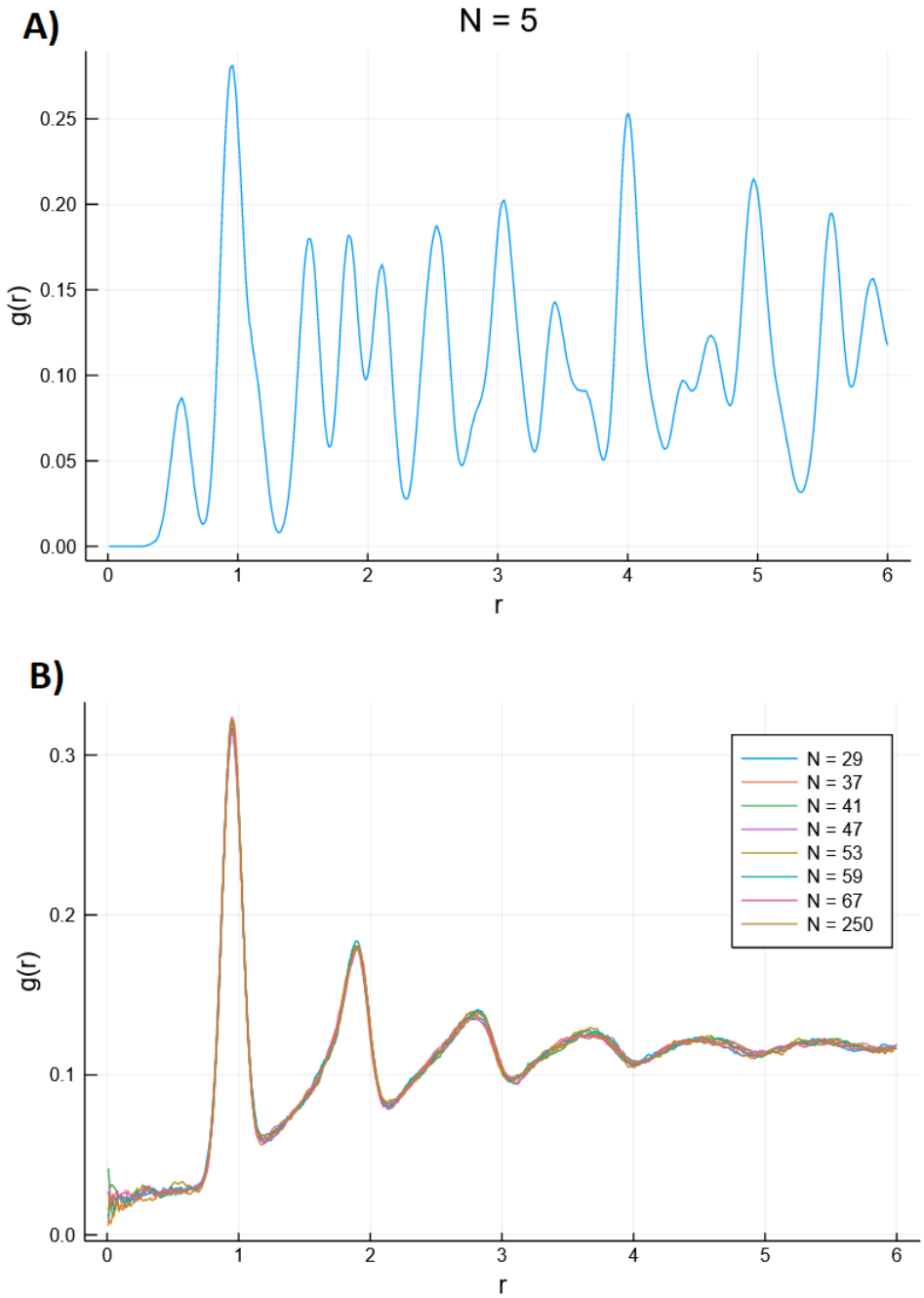

Figura 6.8: Funciones de distribución radial $g(r)$. La imagen **A** corresponde a un arreglo cuasiperiódico de simetría pentagonal. La imagen **B** corresponde a las gráficas obtenidas para los arreglos cuasiperiódicos con simetría rotacional N, donde $N \in \{29, 37, 41, 47, 53, 59, 67, 250\}$.

Capítulo 7

Conclusiones

Se desarrolló un algoritmo eficiente basado en el método dual generalizado con el cual es posible obtener vecindades de un arreglo cuasiperiódico de simetría rotacional N, con $N \in \mathbb{Z}^+$, alrededor de un punto arbitrario en el plano. Como resultado principal de esta tesis, se generó un código en el lenguaje de programación Julia con la implementación de dicho algoritmo, este se encuentra disponible para su descarga en https://github.com/AlanRodrigoMendozaSosa/Quasiperiodic-Tiles, junto con una exhaustiva documentación que incluye diversos notebooks de Jupyter que ejemplifican las diferentes etapas del proceso de creación del código final, así como algunas aplicaciones.

Con respecto a la eficiencia, se realizó una prueba para comparar el tiempo de cómputo requerido en generar y encontrar los vértices que conforman al polígono de un arreglo cuasiperiódico con simetría pentagonal que contiene en su interior a un punto P arbitrario cuya distancia al origen es r, empleando nuestro algoritmo y un algoritmo ingenuo. Se encontró que el tiempo de cómputo requerido por nuestro algoritmo es una constante, independientemente del valor de r, mientras que para el caso del algoritmo ingenuo, este sigue una ley de potencias con exponente 2.377 al ir aumentando la distancia al origen del punto P. Cabe la pena mencionar el uso de la RAM en ambos casos, manteniéndose en el orden de los 500 kB al implementar nuestro algoritmo (independientemente de la distancia al origen del punto P), mientras que al hacer uso del algoritmo ingenuo, esta escalaba rápidamente al orden de los 10 GB conforme aumentaban los valores del parámetro r.

Como una de las aplicaciones de nuestro código, se midió la difusión presente en un sistema cuasiperiódico con simetría pentagonal de una partícula puntual bajo un potencial de interacción de esferas duras entre dicha partícula y unos discos de radio $r = 0.05$ unidades. Este sistema presentó una difusión normal al analizar el desplazamiento cuadrático medio de la partícula como función del tiempo de vuelo, obteniéndose una gráfica cuyo comportamiento es descrito por la ecuación $y = 0.854x^{1.01}$.

Por último, se analizó la función de distribución radial $g(r)$ para múltiples arreglos cuasiperiódicos al ir aumentando la simetría rotacional de estos. Los casos analizados corresponden a una simetría N donde $N \in \{5, 29, 37, 41, 47, 53, 59, 67, 250\}$. Se observó que, para el caso $N = 5$, la función de distribución radial presenta varios picos bien definidos que no parecen desaparecer conforme el parámetro r aumenta, mientras que en el resto de casos la gráfica $g(r)$ presenta dos picos bien definidos en los valores cercanos a $r = 1$ y $r = 2$ unidades, tendiendo a un valor constante cercanos a 0.1 conforme el parámetro r incrementa. Esto nos indica que a simetrías bajas, los arreglos cuasiperiódicos poseen una función de distribución radial similar a la de los cristales, mientras que para simetrías altas, estos tienen un comportamiento similar a la de los sistemas desordenados.

Apéndice A

Pseudocódigo

A continuación se muestra el pseudocódigo de las principales funciones que conforman los algoritmos descritos en el capítulo 4. El código se desarrolló en el lenguaje de programación "Julia" en su versión 1.2.0 y está disponible para su descarga (junto con una documentación más extensa y varios ejemplos) en https://github.com/AlanRodrigoMendozaSosa/Quasiperiodic-Tiles. La parte correspondiente a generar teselados de Voronoi en dos dimensiones fue desarrollada por Enrique Gómez Cruz y la documentación pertinente está disponible en https://github.com/couscouscricket/VoronoiDiagram.

Para diferenciar a las variables con múltiples entradas (como los arreglos o los diccionarios) de aquellas variables con una única entrada (como el índice de alguna iteración) hemos escrito a las primeras en negritas. Por ejemplo, en el primer algoritmo, la salida de la función **X** es un arreglo con las coordenadas asociadas al eje X de los sitios de la vecindad del arreglo cuasiperiódico que se genera dentro de dicha función (es decir, tiene tantas entradas como número de sitios tiene la vecindad), mientras que la primera entrada de la función, β, no está escrita en negritas al ser dicha variable un número entero.

Algorithm 1 parche_Cuadrado

Input
- β Margen de error de los números enteros n_i
- I_V Iteraciones para eliminar clústers aislados
- A Área para detectar polígonos exteriores en los clústers
- R_A Radio del círculo que aproxima a las vecindades
- γ Factor que escala a R_A para generar R_S
- **D** Distancias entre las "franjas ortogonales"
- **S** Vectores estrella
- $\mathbf{A}_\alpha$ Constantes α_i asociadas a cada vector estrella
- **P** Punto alrededor del cual se genera la vecindad

Output
    X Coordenadas X de los sitios de la vecindad
    Y Coordenadas Y de los sitios de la vecindad
procedure GENERAR UN PARCHE CUADRADO CENTRADO EN **P**
    Definir los vértices de un cuadrado.
    $\mathbf{V}_1 \leftarrow [1, 1]$; ▷ 1era esquina
    $\mathbf{V}_2 \leftarrow [-1, 1]$; ▷ 2da esquina
    $\mathbf{V}_3 \leftarrow [-1, -1]$; ▷ 3era esquina
    $\mathbf{V}_4 \leftarrow [1, -1]$; ▷ 4ta esquina
    Generar los puntos $\mathbf{C}_i$.
    $\mathbf{C}_1 \leftarrow \mathbf{P} + \frac{\sqrt{2}}{2}\gamma\ R_A\mathbf{V}_1$; ▷ 1er centro
    $\mathbf{C}_2 \leftarrow \mathbf{P} + \frac{\sqrt{2}}{2}\gamma\ R_A\mathbf{V}_2$; ▷ 2do centro
    $\mathbf{C}_3 \leftarrow \mathbf{P} + \frac{\sqrt{2}}{2}\gamma\ R_A\mathbf{V}_3$; ▷ 3er centro
    $\mathbf{C}_4 \leftarrow \mathbf{P} + \frac{\sqrt{2}}{2}\gamma\ R_A\mathbf{V}_4$; ▷ 4to centro
    for i in $1 : 4$ **do**
        Generar una vecindad alrededor de $\mathbf{C}_i$.
        $\mathbf{Pts}_i \leftarrow$ region_Local_Voronoi$(\beta, \mathbf{D}, \mathbf{S}, \mathbf{A}_\alpha, \mathbf{C}_i)$;
        Obtener centroides $\mathbf{Ce}_i$ *y diccionario* $\mathbf{D}_{Ce_i}$ *que relaciona*
        Centroide $\longrightarrow$ *Vértices de los polígonos de* $\mathbf{V}_i$.
        $\mathbf{Ce}_i, \mathbf{D}_{Ce_i} \leftarrow$ centroides$(\mathbf{Pts}_i)$;
        Obtener el clúster central de la vecindad generada.
        $\mathbf{Ce}_i \leftarrow$ centroides_Area_Acotada_Iterada$(\mathbf{Ce}_i, A, I_V)$;
    end for
    Unir los centroides de las cuatro vecindades.
    $\mathbf{Ce} \leftarrow \bigcup_{i=1}^{4} \mathbf{Ce}_i$;
    Eliminar los centroides repetidos.
    $\mathbf{Ce} \leftarrow$ unique! $(\mathbf{Ce})$;
    Unir los diccionarios de las cuatro vecindades.
    $\mathbf{D}_{Ce} \leftarrow \bigcup_{i=1}^{4} \mathbf{D}_{Ce_i}$;
    Obtener las coordenadas de los vértices a partir de los centroides.
    $\mathbf{X}, \mathbf{Y} \leftarrow$ centroides_A_Vertices$(\mathbf{Ce}, \mathbf{D}_{Ce})$;
    return X, Y

Algorithm 2 region_Local_Voronoi

Input
    β Margen de error de los números enteros n_i
    D Distancias entre las "franjas ortogonales"
    S Vectores estrella
    $\mathbf{A}_\alpha$ Constantes α_i asociadas a cada vector estrella
    P Punto alrededor del cual se genera la vecindad

Output
    Pts Coordenadas (X, Y) de los sitios de la vecindad
procedure GENERAR VECINDAD "CIRCULAR" CENTRADA EN **P**
    Aproximar los números enteros n_i asociados a los vectores estrella $\mathbf{e}_i$.
    Proj $\leftarrow$ proyecciones_Pto_Direccion_Franjas $(\mathbf{P}, \mathbf{D}, \mathbf{S})$;
    Generar la vecindad con las combinaciones (n_i, n_j) de $(\mathbf{e}_i, \mathbf{e}_j)$.
    Pts $\leftarrow$ generador_Vecindades_Vertices $(\mathbf{Proj}, \mathbf{S}, \mathbf{A}_\alpha, \beta)$;
    return Pts

Algorithm 3 proyecciones_Pto_Direccion_Franjas

Input
    P Punto alrededor del cual se genera la vecindad
    D Distancias entre las "franjas ortogonales"
    S Vectores estrella
Output
    Proj Arreglo de números enteros n_i
procedure APROXIMAR LOS NÚMEROS ENTEROS n_i ASOCIADOS A LOS VECTORES ESTRELLA $\mathbf{e}_i$ QUE FORMAN AL POLÍGONO QUE CONTIENE AL PUNTO **P**
    Definir el arreglo que contendrá los enteros n_i
    Proj $\leftarrow$ [ ] ;
    Iterar sobre los vectores estrella
    for i in 1 : length $(\mathbf{S})$ **do**
        $\mathbf{e}_i \leftarrow \mathbf{S}[i]$; ▷ i-ésimo vector estrella
        $d_i \leftarrow \mathbf{D}[i]$; ▷ distancia franjas dirección $\mathbf{e}_i$
        $n_i \leftarrow (\mathbf{e}_i \cdot \mathbf{P}) / d_i$;
        push! $(\mathbf{Proj}, n_i)$;
    end for
    return Proj

Algorithm 4 generador_Vecindades_Vertices

Input
    Proj Arreglo de números enteros n_i
    S Vectores estrella
    $\mathbf{A}_\alpha$ Constantes α_i asociadas a cada vector estrella
    β Margen de error de los números enteros n_i
Output
    Pts Coordenadas (X, Y) de los sitios de la vecindad
procedure GENERAR LOS SITIOS DEL ARREGLO CUASIPERIÓDICO FORMADO LOS ENTEROS n_i EN LA EXPRESIÓN DE NAUMIS-ARAGÓN.
    Definir el arreglo que contendrá los vértices
    Pts $\leftarrow$ [ ] ;

Iterar sobre los vectores estrella
for i in $1 : \text{length}\,(\mathbf{S})$ **do**
  for j in $(i+1) : \text{length}\,(\mathbf{S})$ **do**
    Iterar sobre el margen de error
    for n in $-\beta : \beta$ **do**
      for m in $-\beta : \beta$ **do**
        Definir los enteros prueba Z_i, Z_j
        $Z_i \leftarrow \lfloor \mathbf{Proj}\,[i] \rceil + n$;
        $Z_j \leftarrow \lfloor \mathbf{Proj}\,[j] \rceil + m$;
        Obtener los vértices $t^0_{n_j,n_k}$, $t^1_{n_j,n_k}$, $t^2_{n_j,n_k}$, $t^3_{n_j,n_k}$
        try
          $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3 \leftarrow \text{cuatro\_Regiones}\,(i, j, Z_i, Z_j, \mathbf{S}, \mathbf{A}_\alpha)$;
        catch *Los vectores estrella son colineales*
          nothing
        end try
        Agregar los vértices a **Pts**
        $\text{push!}\,(\mathbf{Pts}, \mathbf{t}_0)$;
        $\text{push!}\,(\mathbf{Pts}, \mathbf{t}_1)$;
        $\text{push!}\,(\mathbf{Pts}, \mathbf{t}_2)$;
        $\text{push!}\,(\mathbf{Pts}, \mathbf{t}_3)$;
      end for
    end for
  end for
end for
return Pts

Algorithm 5 cuatro_Regiones

Input
  J Índice del vector estrella a considerar
  K Índice del vector estrella a considerar
  n_j Número entero asociado al vector estrella $\mathbf{e}_j$
  n_k Número entero asociado al vector estrella $\mathbf{e}_k$
  $\mathbf{S}$ Vectores estrella
  $\mathbf{A}_\alpha$ Constantes α_i asociadas a cada vector estrella
Output
  $\mathbf{t}_0$ Vértice $t^0_{n_j,n_k}$ de un polígono
  $\mathbf{t}_1$ Vértice $t^1_{n_j,n_k}$ de un polígono
  $\mathbf{t}_2$ Vértice $t^2_{n_j,n_k}$ de un polígono
  $\mathbf{t}_3$ Vértice $t^3_{n_j,n_k}$ de un polígono
procedure GENERAR LOS CUATRO VÉRTICES DE UN POLÍGONO DEL ARREGLO CUASIPERIÓDICO

Revisar si $\mathbf{e}_j$ *y* $\mathbf{e}_k$ *son colineales.*
if $\operatorname{mod}(\operatorname{length}(\mathbf{S}), 2) = 0$ **AND** $K = J + \operatorname{length}(\mathbf{S})/2$ **then**
    error: "Los vectores $\mathbf{e}_j$ *y* $\mathbf{e}_k$ *son colineales."*
end if
Definir los vectores estrella $\mathbf{e}_j$ *y* $\mathbf{e}_k$*.*
$\mathbf{e}_j \leftarrow \mathbf{S}[J]$;
$\mathbf{e}_k \leftarrow \mathbf{S}[K]$;
Definir los vectores ortogonales a $\mathbf{e}_j$ *y* $\mathbf{e}_k$*.*
$\mathbf{e}_j^{\perp} \leftarrow \text{vector\_Ortogonal}(\mathbf{e}_j)$;
$\mathbf{e}_k^{\perp} \leftarrow \text{vector\_Ortogonal}(\mathbf{e}_k)$;
Definir los parámetros para las rectas ortogonales a $\mathbf{e}_j$ *y* $\mathbf{e}_k$*.*
$P_{e_j} \leftarrow n_j + \mathbf{A}_\alpha[J]$;
$P_{e_k} \leftarrow n_k + \mathbf{A}_\alpha[K]$;
Obtener el área formada por los vectores $\mathbf{e}_j$ *y* $\mathbf{e}_k$*.*
$e_{jx} \leftarrow \mathbf{e}_j[1]$; ▷ Componente x de $\mathbf{e}_j$
$e_{jy} \leftarrow \mathbf{e}_j[2]$; ▷ Componente y de $\mathbf{e}_j$
$e_{kx} \leftarrow \mathbf{e}_k[1]$; ▷ Componente x de $\mathbf{e}_k$
$e_{ky} \leftarrow \mathbf{e}_k[2]$; ▷ Componente y de $\mathbf{e}_k$
$A_{JK} \leftarrow e_{jx} * e_{ky} - e_{jy} * e_{kx}$;
Definir el vértice $t^0_{n_j,n_k}$*.*
$\mathbf{t}_0 \leftarrow n_j\mathbf{e}_j + n_k\mathbf{e}_k$;
Iterar sobre los vectores estrella excepto $\mathbf{e}_j$ *y* $\mathbf{e}_k$*.*
for i in $1 : \operatorname{length}(\mathbf{S})$ **do**
    if $i \neq J$ **AND** $i \neq K$ **then**
        $\mathbf{e}_i \leftarrow \mathbf{S}[i]$; ▷ i-ésimo vector estrella
        $\alpha_i \leftarrow \mathbf{A}_\alpha[i]$; ▷ constante α_i asociada a $\mathbf{e}_i$
        $P_{e_i} \leftarrow \frac{P_{e_j}}{A_{JK}}\left(\mathbf{e}_k^{\perp} \cdot \mathbf{e}_i\right) - \frac{P_{e_k}}{A_{JK}}\left(\mathbf{e}_j^{\perp} \cdot \mathbf{e}_i\right)$;
        $\mathbf{t}_0 \leftarrow \mathbf{t}_0 + \lfloor P_{e_i} - \alpha_i \rfloor * \mathbf{e}_i$;
    end if
end for
Generar los vértices $t^1_{n_j,n_k}$, $t^2_{n_j,n_k}$, $t^3_{n_j,n_k}$*.*
$\mathbf{t}_1 \leftarrow \mathbf{t}_0 - \mathbf{e}_j$;
$\mathbf{t}_2 \leftarrow \mathbf{t}_0 - \mathbf{e}_j - \mathbf{e}_k$;
$\mathbf{t}_3 \leftarrow \mathbf{t}_0 - \mathbf{e}_k$;
return $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$

Algorithm 6 centroides

Input
    Pts Coordenadas (X, Y) de los sitios de la vecindad
Output
    Ce Coordenadas (X, Y) de los centroides
    $\mathbf{D_{Ce}}$ Diccionario Centroide $\longrightarrow$ Vertices

procedure GENERAR LOS CENTROIDES DE LOS POLÍGONOS DEL ARREGLO CUASIPERIÓDICO
    Definir el arreglo para las coordenadas de los centroides.
    $\mathbf{Ce} \leftarrow [\,]$;
    Definir el diccionario Centroides $\longrightarrow$ *Vertices.*
    $\mathbf{D}_{Ce} \leftarrow \text{Dict}()$;
    Iterar sobre cada polígono
    for i in $1 : 4 : (\text{length}(\mathbf{Pts}) - 3)$ **do**
        Obtener vértices polígono i-ésimo.
        $\mathbf{V1}_i \leftarrow \mathbf{Pts}\,[i]$; ▷ 1er Vértice
        $\mathbf{V2}_i \leftarrow \mathbf{Pts}\,[i+1]$; ▷ 2do Vértice
        $\mathbf{V3}_i \leftarrow \mathbf{Pts}\,[i+2]$; ▷ 3er Vértice
        $\mathbf{V4}_i \leftarrow \mathbf{Pts}\,[i+3]$; ▷ 4to Vértice
        Obtener las coordenadas del centroide.
        $\mathbf{Ce}_X \leftarrow \frac{1}{4}\sum_{k=1}^{4} \mathbf{Vk}_i\,[1]$; ▷ Componente x
        $\mathbf{Ce}_Y \leftarrow \frac{1}{4}\sum_{k=1}^{4} \mathbf{Vk}_i\,[2]$; ▷ Componente y
        Agregar el centroide i-ésimo al arreglo para centroides.
        $\text{push!}\,(\mathbf{Ce}, (\mathbf{Ce}_X, \mathbf{Ce}_Y))$;
        Asociar el centroide con los vértices del polígono.
        $\mathbf{D}_{Ce}\,[(\mathbf{Ce}_X, \mathbf{Ce}_Y)] \leftarrow (\mathbf{V1}_i, \mathbf{V2}_i, \mathbf{V3}_i, \mathbf{V4}_i)$;
    end for
    return $\mathbf{Ce}, \mathbf{D}_{Ce}$

Algorithm 7 centroides_Area_Acotada_Iterada

Input
    Ce Coordenadas (X, Y) de los centroides
    A Área para detectar polígonos exteriores en los clústers
    I_V Iteraciones del algoritmo
Output
    Ce Coordenadas (X, Y) de los centroides del clúster principal
procedure ELIMINAR LOS CLÚSTERS Y POLÍGONOS AISLADOS
    for i in $1 : I_V$ **do**
        Generar la estructura del teselado de Voronoi.
        $\mathbf{Vor} \leftarrow \text{getVoronoiDiagram}\,(\mathbf{Ce})$; ▷ Voronoi_Code
        Eliminar la capa exterior de los clústers.
        $\mathbf{Ce} \leftarrow \text{centroides\_Area\_Acotada}\,(\mathbf{Vor}, A)$;
    end for
    return Ce

Algorithm 8 centroides_Area_Acotada

Input
  Vor Estructura de la teselación de Voronoi
  A Área para detectar polígonos exteriores en los clústers
Output
  Ce Coordenadas (X, Y) de los centroides
procedure ELIMINAR LA CAPA EXTERIOR DE TODOS LOS CLÚSTERS
  Definir el arreglo para los centroides.
  Ce $\leftarrow [\,]$;
  Iterar sobre todos los centros de Voronoi.
  for Face in **Vor**.faces **do** ▷ Conjunto de centros
    Revisar el área de la celda de Voronoi.
    if Face.area $< A$ **then**
      push! (**Ce**, Face.site) ;
    end if
  end for
  return Ce

Algorithm 9 centroides_A_Vertices

Input
  Ce Coordenadas (X, Y) de los centroides
  $\mathbf{D}_{Ce}$ Diccionario Centroide $\longrightarrow$ Vértices
Output
  X Coordenadas X de los sitios de la vecindad
  Y Coordenadas Y de los sitios de la vecindad
procedure OBTENER LAS COORDENADAS DE LOS SITIOS DE LA VECINDAD DEL ARREGLO CUASIPERIÓDICO
  Definir los arreglos que contendrán las coordenadas X, Y
  $X \leftarrow [\,]$;
  $Y \leftarrow [\,]$;
  Iterar sobre todos los centroides
  for i in **Ce do**
    Coordenadas de los vértices del polígono
    $V_{1x}, V_{2x}, V_{3x}, V_{4x} \leftarrow (\mathbf{D}_{Ce}\,[i])\,[1]$; ▷ componentes x
    $V_{1y}, V_{2y}, V_{3y}, V_{4y} \leftarrow (\mathbf{D}_{Ce}\,[i])\,[2]$; ▷ compomentes y
    push! $(\mathbf{X}, [V_{1x}, V_{2x}, V_{3x}, V_{4x}])$;
    push! $(\mathbf{Y}, [V_{1y}, V_{2y}, V_{3y}, V_{4y}])$;
  end for
  Aplanar los arreglos $\mathbf{X}, \mathbf{Y}$ *antes de regresarlos*
  return X, Y

Algorithm 10 MSD_Varying_Velocities

Input
- β Margen de error de los números enteros n_i
- I_V Iteraciones para eliminar clústers aislados
- A Área para detectar polígonos exteriores en los clústers
- R_A Radio del círculo que aproxima a las vecindades
- γ Factor que escala a R_A para generar R_S
- $\mathbf{D}$ Distancias entre las "franjas ortogonales"
- $\mathbf{S}$ Vectores estrella
- $\mathbf{A}_\alpha$ Constantes α_i asociadas a cada vector estrella
- $\mathbf{P}$ Punto alrededor del cual se genera la vecindad
- N_V Número de velocidades iniciales
- N_{sf} Número de vuelos parciales
- T_{sf} Tiempo de vuelo parcial
- $\mathbf{A}_M$ Desplazamiento Cuadrático Medio
- F_{CC} Función para el cambio de celda
- F_{CS} Función para actualizar condiciones de la partícula

Output
- $\mathbf{A}_M$ Desplazamiento Cuadrático Medio
- N_f Número de vuelos completados

procedure CALCULAR EL DESPLAZAMIENTO CUADRÁTICO MEDIO DE UN ARREGLO CUASIPERIÓDICO A PARTIR DE UNA POSICIÓN INICIAL FIJA

Generar un parche cuadrado centrado en $\mathbf{P}$

$\mathbf{X}, \mathbf{Y} \leftarrow \text{parche\_Cuadrado}\left(\beta, I_V, A, R_A, \gamma, \mathbf{D}, \mathbf{S}, \mathbf{A}_\alpha, \mathbf{P}\right);$

Obtener las coordenadas (X, Y) de los sitios del arreglo cuasiperiódico

$\mathbf{Pts}_V \leftarrow [\,];$

for i in $1 : \text{length}(\mathbf{X})$ **do**

$\quad \text{push!}\left(\mathbf{Pts}_V, \left(\mathbf{X}\left[i\right], \mathbf{Y}\left[i\right]\right)\right);$

end for

$\mathbf{Pts}_V \leftarrow \text{unique!}\left(\mathbf{Pts}_V\right);$ ▷ Eliminar sitios repetidos

Determinar qué sitios pertenecen a la región cuadrada

$\mathbf{D}_{VC} \leftarrow \text{Dict}\left(\,\right);$

$R_S \leftarrow \gamma * R_A;$ ▷ Radio seguro vecindad "circular"

for i in $\mathbf{Pts}_V$ **do**

$\quad$ **if** Condición_1 **AND** Condición_2 **then**

$\quad\quad \mathbf{D}_{VC}\left[i\right] \leftarrow \text{true};$

$\quad$ **else**

$\quad\quad \mathbf{D}_{VC}\left[i\right] \leftarrow \text{false};$

$\quad$ **end if**

end for

Condición_1: $\mathbf{P}\left[1\right] - \sqrt{2}R_S \leq i\left[1\right] \leq \mathbf{P}\left[1\right] + \sqrt{2}R_S;$

Condición_2: $\mathbf{P}\left[2\right] - \sqrt{2}R_S \leq i\left[2\right] \leq \mathbf{P}\left[2\right] + \sqrt{2}R_S;$

```
41:  Generar el teselado Voronoi con los sitios del parche cuadrado
42:  Vor_V ← getVoronoiDiagram (Pts_V) ;
43:  Generar diccionario: Sitio ⟶ índice Voronoi
44:  D_VI ← diccionario_Centroides_Indice_Voronoi (Pts_V, Vor_V) ;
45:  Generar un arreglo para los centros de los parches cuadrados
46:  Ctrs_C ← [P] ;
47:  Encontrar la celda de Voronoi que contiene a la partícula
48:  PP ← copy (P) ;                                  ▷ Copia de la posición inicial
49:  Vecinos ← vertices_Cercanos_Voronoi (PP, Pts_V) ;
50:  Cercano ← vertice_Cercano (Vecinos, PP) ;
51:  Iterar sobre el número de velocidades
52:  N_f ← 0;                          ▷ Contador del número de vuelos completados
53:  for i in 1 : N_V do
54:      Close ← false;
55:      Generar una velocidad inicial
56:      θ ← 2 ∗ π ∗ rand () ;
57:      Vel ← [cos (θ) , sin (θ)] ;
58:      VVel ← copy (Vel) ;                       ▷ Copia de la velocidad inicial
59:      PP ← copy (P) ;                           ▷ Copia de la posición inicial
60:      Obtener índice de Voronoi de la celda contenedora
61:      I_Vor ← D_VI [Cercano] ;
62:      Door ← [[Inf, Inf] , [Inf, Inf]]
63:      Iterar sobre el número de vuelos parciales
64:      for j in 1 : N_sf do
65:          Definir tiempo de vuelo
66:          T_RF ← T_sf;                          ▷ Tiempo de vuelo restante
67:          try
68:              Calcular la trayectoria de la partícula
69:              Arg0 ← vuelo_Una_Particula_Obstaculos (Arg1, Arg2) ;
70:              Arg0: PP, VVel, T_RF, I_Vor, Door;
71:              Arg1: PP, VVel, T_sf, I_Vor, Vor_V;
72:              Arg2: D_VI, D_VC, Door, F_CC, F_CS;
73:          catch Err
74:              if typeof (Err) = DeadTrajectory then
75:                  Close ← true;                 ▷ Trajectoria cerrada
76:              else
77:                  throw (Error) ;               ▷ Error cálculo trayectoria
78:              end if
79:          end try
80:          Obtener el vértice más cercano a la partícula
81:          Cercano_F ← Vor_V.faces [I_Vor] .site;
82:          while T_RF > 0 do
83:              La partícula se salió de las regiones cuadradas
84:              Generar nuevo parche cuadrado
```

85: $\mathbf{X}, \mathbf{Y} \leftarrow$ parche_Cuadrado $(\beta, I_V, A, R_A, \gamma, \mathbf{D}, \mathbf{S}, \mathbf{A}_\alpha, \mathbf{PP})$;
86: *Agregar centro del nuevo parche cuadrado a su arreglo*
87: push!$(\mathbf{Ctrs}_C, \mathbf{PP})$;
88: *Obtener las coordenadas de los sitios del parche cuadrado*
89: $\mathbf{Pts}_{NV} \leftarrow [\,]$; ▷ Arreglo nuevos sitios
90: **for** i in 1 : length($\mathbf{X}$) **do**
91: push! $(\mathbf{Pts}_{NV}, (\mathbf{X}[i], \mathbf{Y}[i]))$;
92: **end for**
93: *Unir sitios de los parches cuadrados*
94: $\mathbf{Pts}_{NV} \leftarrow$ unique! $(\mathbf{Pts}_{NV})$; ▷ Eliminar repetidos
95: $\mathbf{Pts}_V \leftarrow \mathbf{Pts}_V \bigcup \mathbf{Pts}_{NV}$;
96: $\mathbf{Pts}_V \leftarrow$ unique! $(\mathbf{Pts}_V)$; ▷ Eliminar repetidos
97: *Determinar qué sitios están dentro de regiones cuadradas*
98: $\mathbf{D}_{VC} \leftarrow$ Dict ( ) ;
99: **for** k in $\mathbf{Pts}_V$ **do**
100: **for** l in $\mathbf{Ctrs}_C$ **do**
101: **if** Condición_1 **AND** Condición_2 **then**
102: $\mathbf{D}_{VC}[i] \leftarrow$ true; **break**
103: **else**
104: $\mathbf{D}_{VC}[i] \leftarrow$ false;
105: **end if**
106: **end for**
107: **end for**
108: Condición_1: $l[1] - \sqrt{2}R_S \leq k[1] \leq l[1] + \sqrt{2}R_S$;
109: Condición_2: $l[2] - \sqrt{2}R_S \leq k[2] \leq l[2] + \sqrt{2}R_S$;
110: *Generar el teselado de Voronoi*
111: $\mathbf{Vor}_V \leftarrow$ getVoronoiDiagram $(\mathbf{Pts}_V)$;
112: *Generar diccionario: Sitio $\longrightarrow$ índice Voronoi*
113: $\mathbf{D}_{VI} \leftarrow$ diccionario_Centroides_Indice_Voronoi (Arg) ;
114: Arg: $\mathbf{Pts}_V, \mathbf{Vor}_V$;
115: *Encontrar el índice de la celda contenedora*
116: $I_{Vor} \leftarrow \mathbf{D}_{VI}[\mathbf{Cercano}_F]$;
117: *Calcular la trayectoria de la partícula*
118: Arg0 $\leftarrow$ vuelo_Una_Particula_Obstaculos (Arg1, Arg2) ;
119: Arg0: $\mathbf{PP}, \mathbf{VVel}, T_{RF}, I_{Vor}, \mathbf{Door}$;
120: Arg1: $\mathbf{PP}, \mathbf{VVel}, T_{RF}, I_{Vor}, \mathbf{Vor}_V$;
121: Arg2: $\mathbf{D}_{VI}, \mathbf{D}_{VC}, \mathbf{Door}, F_{CC}, F_{CS}$;
122: *Obtener el vértice más cercano a la partícula*
123: $\mathbf{Cercano}_F \leftarrow \mathbf{Vor}_V.faces[I_{Vor}].site$;
124: **end while**
125: *Actualizar el MSD del sistema*
126: $PP_x \leftarrow \mathbf{PP}[1]$; $PP_y \leftarrow \mathbf{PP}[2]$;
127: $P_x \leftarrow \mathbf{P}[1]$; $P_y \leftarrow \mathbf{P}[2]$;

```
            MSD ← (PP_x − P_x)^2 + (PP_y − P_y)^2 ;
            A_M [j] ← (A_M [j] ∗ N_f + MSD) / (N_f + 1) ;
        end for
        Contabilizar si la trayectoria finalizó correctamente
        if Close = false then
            N_f = N_f + 1;
        end if
    end for
    return A_M, N_f
```

Algorithm 11 diccionario_Centroides_Indice_Voronoi

Input
  Pts Coordenadas (X, Y) de los centros de Voronoi
  Vor Estructura de la teselación de Voronoi
Output
  $\mathbf{D_{CI}}$ Diccionario Centro $\longrightarrow$ Indice Voronoi
procedure Generar un diccionario que dado el centro de Voronoi regrese el índice dentro de la estructura de Voronoi asociado a la celda de Voronoi que generó dicho centro
  $\mathbf{D}_{CI} \leftarrow$ Dict ( ) ;
  for i in 1 : length (**Pts**) **do**
    $\mathbf{D}_{CI}$ [**Vor**.$faces$ [i] .$site$] $\leftarrow i$;
  end for
  return $\mathbf{D}_{CI}$

Algorithm 12 vertices_Cercanos_Voronoi

Input
  P Punto al que queremos encontrarle sus primeros vecinos
  Pts Coordenadas (X,Y) de los sitios del arreglo cuasiperiódico
Output
  Vec Coordenadas (X,Y) de los primeros vecinos de **P**
procedure Dado un punto **P** en el espacio y un conjunto de puntos de un arreglo cuasiperiódico, regresa los puntos más cercanos a **P**
  Agregamos el punto **P** *al conjunto de puntos del arreglo cuasiperiódico*
  push! (**Pts**, **P**) ;
  Generar la teselación de Voronoi
  Vor ← getVoronoiDiagram (**Pts**) ;
  Obtener el índice de la celda de Voronoi asociado a **P**
  $I_P \leftarrow$ indice_Voronoi_Centroide (**P**, **Vor**) ;
  Obtener los vecinos a la celda de Voronoi asociado a **P**
  Vec ← vecinos_Voronoi (I_P, **Vor**) ;

```
15:     Eliminar al punto P del conjunto de puntos
16:     pop!(Pts) ;                                  ▷ Elimina el último elemento
        return Vec
```

Algorithm 13 indice_Voronoi_Centroide

```
 1: Input
 2:     P  Centro de Voronoi
 3:     Vor Estructura de la teselación de Voronoi
 4: Output
 5:     I_P  Indice de la celda de Voronoi asociada a P
 6: procedure Obtener el índice, dentro de la estructura del teselado
    de Voronoi, de la celda de Voronoi asociado a P
 7:     I_P ← 1;                                     ▷ Indice para iterar sobre las celdas
 8:     Iterar sobre los celdas de Voronoi
 9:     while P ≠ Vor.faces[I_P].site do
10:         I_P ← I_P + 1;
11:     end while
        return I_P
```

Algorithm 14 vecinos_Voronoi

```
 1: Input
 2:     I_V  Índice de la celda de Voronoi
 3:     Vor Estructura de la teselación de Voronoi
 4: Output
 5:     Vec Coordenadas (X,Y) de los primeros vecinos de P
 6: procedure Obtener las celdas de Voronoi vecinas a la celda con índice
    I_V
 7:     Definir el arreglo donde irán las celdas vecinas
 8:     Vec ← [ ] ;
 9:     Tomar un lado arbitrario de la celda con índice I_V
10:     Seg ← Vor.faces[I_V].outerComponent;
11:     Iterar sobre los lados de la celda
12:     while true do
13:         Obtener las coordenadas del centro de Voronoi de la celda vecina
14:         Pt ← Seg.twin.incidentFace.site;
15:         push!(Vec, Pt) ;
16:         Recorrer al siguiente lado de la celda
17:         Seg ← Seg.next;
```

```
        Verificar si el nuevo lado es el lado inicial
        if Seg = Vor.faces[I_V].outerComponent then
            break
        end if
    end while
     return Vec
```

Algorithm 15 vertice_Cercano

```
Input
    Vec Sitios del arreglo cuasiperiódico cercanos a un punto de interés
    P   Punto de interés en el espacio 2D
Output
    Near  Sitio más cercano a P
procedure Obtener el sitio más cercano a un punto de interés dado un
conjunto de puntos de un arreglo cuasiperiódico
    Obtener el número de sitios a considerar
    N ← length(Vec);
    Definir el arreglo que contendrá las distancia punto-sitio
    Dist ← [ ];
    Definir diccionario: Distancia ⟶ Indice
    D_DI ← Dict( );
    Iterar sobre los sitios
    for i in 1 : N do
        Near ← Vec[i];                          ▷ Sitio candidato
        L ← |Near − P|;
        push!(Dist, L);
        D_DI[L] ← i;
    end for
    Seleccionar el sitio más cercano
    m ← minimum(Dist);                          ▷ Distancia mínima
    j ← D_DI[m];                                ▷ Índice del sitio más cercano
    Near ← Vec[j];
     return Near
```

Algorithm 16 vuelo_Una_Particula_Obstaculos

```
Input
    P   Posición de la partícula
    V   Velocidad de la partícula
    T_F  Tiempo de vuelo de la partícula
    I_C  Índice de la celda de Voronoi contenedora
    Vor Estructura de la teselación de Voronoi
    D_VI Diccionario Sitios ⟶ Índices
```

8: $\mathbf{D}_{VC}$ Diccionario Sitios $\longrightarrow$ Vecindad Cuadrada
9: **Door** Vértices del segmento por el que entró la partícula
10: F_{CC} Función para el cambio de celda
11: F_{CS} Función para actualizar condiciones de la partícula
12: **Output**
13: **P** Posición de la partícula
14: **V** Velocidad de la partícula
15: T_F Tiempo de vuelo de la partícula
16: I_C Índice de la celda de Voronoi contenedora
17: **Door** Vértices del segmente por el que entró la partícula
18: **procedure** CALCULAR LA TRAYECTORIA DE LA PARTÍCULA EN UN INTERVALO DE TIEMPO
19: **while** $T_F > 0$ **do**
20: **Cen** $\leftarrow$ **Vor**.$faces\,[I_C]\,.site$; ▷ Centro de la celda de Voronoi contenedora
21: *Verificar si la celda contenedora está fuera de la región cuadrada*
22: **if** $\mathbf{D}_{VC}\,[\mathbf{Cen}] =$ false **then**
23: **return** $\mathbf{P}, \mathbf{V}, T_F, I_C, \mathbf{Door}$
24: **end if**
25: **PP** $\leftarrow$ copy $(\mathbf{P})$; ▷ Copia posición
26: **VV** $\leftarrow$ copy $(\mathbf{V})$; ▷ Copia velocidad
27: **DDoor** $\leftarrow$ copy $(\mathbf{Door})$; ▷ Copia segmento de entrada
28: $\mathbf{D}_{SI} \leftarrow$ Dict $(\,)$; ▷ Diccionario: Segmento $\longrightarrow$ Indice Vecino
29: $\mathbf{D}_{SC} \leftarrow$ Dict $(\,)$; ▷ Diccionario: Segmento $\longrightarrow$ Centro Vecino
30: $\mathbf{A}_V \leftarrow [\,]$; ▷ Arreglo con los vértices de la celda contenedora
31: *Obtener un segmento de la celda contenedora*
32: **Seg** $\leftarrow$ **Vor**.$faces\,[I_C]\,.outerComponent$;
33: **while** true **do**
34: *Obtener centro de Voronoi de la celda vecina*
35: **Vec** $\leftarrow$ **Seg**.$twin.incidentFace.site$;
36: $I_{Vec} \leftarrow \mathbf{D}_{VI}\,[\mathbf{Vec}]$; ▷ Índice vecino
37: *Obtener los vertices del segmento en turno*
38: $\mathbf{Pt}_1 \leftarrow$ **Seg**.$origin.coordinates$;
39: $\mathbf{Pt}_2 \leftarrow$ **Seg**.$next.origin.coordinates$;
40: push!$(\mathbf{A}_V, \mathbf{Pt}_1)$; ▷ Guardar vértice
41: *Actualizar diccionarios*
42: $\mathbf{D}_{SI}\,[\mathbf{Pt}_1, \mathbf{Pt}_2] \leftarrow I_{Vec}$;
43: $\mathbf{D}_{SC}\,[\mathbf{Pt}_1, \mathbf{Pt}_2] \leftarrow$ **Vec**;
44: **Seg** $\leftarrow$ **Seg**.$next$; ▷ Siguiente segmento
45: **if Seg** $=$ **Vor**.$faces\,[I_C]\,.outerComponent$; **then**
46: **break** ▷ El segmento es el segmento inicial
47: **end if**
48: **end while**

```
49:         Calcular la trayectoria de la partícula
50:         P_C, V_C, T_FC, Door_C ← F_CC (Args) ;
51:         Args: PP, VV, A_V, DDoor, Cen, D_SC;
52:         Obtener índice de la celda receptora
53:         I_PC ← 0;
54:         if T_FC < Inf then
55:             X_1 ← Door_C [1] ;                          ▷ Vértice 1 del segmento
56:             X_2 ← Door_C [2] ;                          ▷ Vértice 2 del segmento
57:             I_PC ← D_SI [X_1, X_2] ;                    ▷ Índice celda receptora
58:         else Es una trayectoria cerrada
59:             throw(DeadTrajectory) ;                     ▷ Arrojamos un error
60:         end if
61:         if T_F − T_FC > 0 then                          ▷ Se cambia de polígono
62:             T_F ← T_F − T_FC;                           ▷ Actualización de tiempo
63:             P ← P_C;                                    ▷ Actualización de posición
64:             V ← V_C;                                    ▷ Actualización de velocidad
65:             I_C ← I_PC;                                 ▷ Actualización de polígono contenedor
66:             Door ← Door_C;                              ▷ Actualización segmento entrada
67:         else No hay cambio de polígono
68:             P, V, Door ← F_CS (P, V, A_V, Door, Cen, D_SC, T_F) ;
69:             return P, V, 0, I_C, Door
70:         end if
71:     end while
```

Bibliografía

[1] D. Shechtman, I. Blech, D. Gratias, and J. W. Cahn, "Metallic phase with long-range orientational order and no translational symmetry," *Physical Review Letters*, vol. 53, pp. 1951–1953, nov 1984. 1, 11

[2] X. Zeng, G. Ungar, Y. Liu, V. Percec, A. E. Dulcey, and J. K. Hobbs, "Supramolecular dendritic liquid quasicrystals," *Nature*, vol. 428, pp. 157–160, mar 2004. 2, 12

[3] T. Dotera, "Quasicrystals in soft matter," *Israel Journal of Chemistry*, vol. 51, pp. 1197–1205, dec 2011. 2, 12

[4] S. Fischer, A. Exner, K. Zielske, J. Perlich, S. Deloudi, W. Steurer, P. Lindner, and S. Förster, "Colloidal quasicrystals with 12-fold and 18-fold diffraction symmetry," *Proceedings of the National Academy of Sciences*, vol. 108, pp. 1810–1814, jan 2011. 2, 12

[5] "Period of a function." Encyclopedia of Mathematics. URL: http://www.encyclopediaofmath.org/index.php?title=Period_of_a_function&oldid=42321. 5

[6] J. C. Hernández, "Funciones cuasi-periódicas de bohr," *Bol. Mat.*, vol. 16, no. 2, pp. 149–165, 2009. 6

[7] Y. V. Komlenko and E. L. Tonkov, "Quasi-periodic function.." Encyclopedia of Mathematics. URL: http://www.encyclopediaofmath.org/index.php?title=Quasi-periodic_function&oldid=14617. 7

[8] "Historia del mosaico." Domus Sophiae Mosaicos. URL: https://domussophiae.com/historia/. 8

[9] M. Gardner, "Mathematical games," *Scientific American*, vol. 236, no. 1, pp. 110–121, 1977. 8, 9, 10, 16

[10] H. Wang, "Proving theorems by pattern recognition - II," *Bell System Technical Journal*, vol. 40, pp. 1–41, jan 1961. 10

[11] C. S. Kaplan, *Introductory Tiling Theory for Computer Graphics*. Morgan & Claypool Publishers, 2009. 10

[12] R. Berger, *The Undecidability of the Domino Problem*. American Mathematical Society, 1966. 10

[13] D. Levine and P. J. Steinhardt, "Quasicrystals. i. definition and structure," *Physical Review B*, vol. 34, pp. 596–616, jul 1986. 10

[14] D. Levine and P. J. Steinhardt, "Quasicrystals: A new class of ordered structures," *Physical Review Letters*, vol. 53, pp. 2477–2480, dec 1984. 10

[15] J. E. S. Socolar, P. J. Steinhardt, and D. Levine, "Quasicrystals with arbitrary orientational symmetry," *Physical Review B*, vol. 32, pp. 5547–5550, oct 1985. 11, 20

[16] N. P. Frank, "A primer of substitution tilings of the euclidean plane," *Expositiones Mathematicae*, vol. 26, no. 4, pp. 295–326, 2008. 11, 14, 16

[17] IUCR *Acta Cryst. A*, vol. 48, p. 922, 1992. http://ww1.iucr.org/comm/capd/terms.html. 12

[18] E. Maciá, "The role of aperiodic order in science and technology," *Reports on Progress in Physics*, vol. 69, pp. 397–441, dec 2005. 12

[19] D. Ratliff, A. Archer, P. Subramanian, and A. Rucklidge, "Which wave numbers determine the thermodynamic stability of soft matter quasicrystals?," *Physical Review Letters*, vol. 123, oct 2019. 13

[20] M. Martinsons and M. Schmiedeberg, "Growth of two-dimensional decagonal colloidal quasicrystals," *Journal of Physics: Condensed Matter*, vol. 30, p. 255403, may 2018. 13

[21] M. Mihalkovič and M. Widom, "Spontaneous formation of thermodynamically stable al-cu-fe icosahedral quasicrystal from realistic atomistic simulations," *Physical Review Research*, vol. 2, feb 2020. 13

[22] M. Martinsons, J. Hielscher, S. C. Kapfer, and M. Schmiedeberg, "Event-chain monte carlo simulations of the liquid to solid transition of two-dimensional decagonal colloidal quasicrystals," *Journal of Physics: Condensed Matter*, vol. 31, p. 475103, aug 2019. 13

[23] Y. S. Chan, C. T. Chan, and Z. Y. Liu, "Photonic band gaps in two dimensional photonic quasicrystals," *Physical Review Letters*, vol. 80, pp. 956–959, feb 1998. 13

[24] Z. Huo, E. Liu, and J. Liu, "Hollow-core photonic quasicrystal fiber with high birefringence and ultra-low nonlinearity," *Chinese Optics Letters*, vol. 18, no. 3, p. 030603, 2020. 13

[25] N. B. Ali, V. Dhasarathan, H. Alsaif, Y. Trabelsi, T. K. Nguyen, Y. Bouazzi, and M. Kanzari, "Design of output-graded narrow polychromatic filter by using photonic quasicrystals," *Physica B: Condensed Matter*, vol. 582, p. 411918, apr 2020. 13

[26] Y. Trabelsi, N. B. Ali, W. Belhadj, and M. Kanzari, "Photonic band gap properties of one-dimensional generalized fibonacci photonic quasicrystal containing superconductor material," *Journal of Superconductivity and Novel Magnetism*, vol. 32, pp. 3541–3547, may 2019. 13

[27] X. Liu and X. Sun, "A vertical cavity surface emitting laser based on fibonacci photon quasicrystal cavity," *Journal of Physics: Conference Series*, vol. 1237, p. 032079, jun 2019. 13

[28] S. V. Boriskina, "Making invisible materials," *Nature Photonics*, vol. 9, pp. 422–424, jun 2015. 13

[29] K. Ueda, T. Dotera, and T. Gemma, "Photonic band structure calculations of two-dimensional archimedean tiling patterns," *Physical Review B*, vol. 75, may 2007. 13

[30] W. Jin, M. Song, X. Yue, and Y. Gao, "Optical induced area-controllable two-dimensional eight-fold symmetric photonic quasicrystal microstructures," *Optical Materials*, vol. 100, p. 109719, feb 2020. 13

[31] J. Mikhael, J. Roth, L. Helden, and C. Bechinger, "Archimedean-like tiling on decagonal quasicrystalline surfaces," *Nature*, vol. 454, pp. 501–504, jul 2008. 13

[32] J. E. S. Socolar, T. C. Lubensky, and P. J. Steinhardt, "Phonons, phasons, and dislocations in quasicrystals," *Physical Review B*, vol. 34, pp. 3345–3360, sep 1986. 14

[33] A. Mendoza, "Movimiento circular en ambientes cuasi-periódicos: magnetorresistencia en cuasicristales," Master's thesis, Universidad Nacional Autónoma de México, 2018. 16

[34] M. Senechal, *Quasicrystals and geometry*. Cambridge University Press, 1996. 17

[35] A. S. Kraemer and D. P. Sanders, "Embedding quasicrystals in a periodic cell: Dynamics in quasiperiodic structures," *Physical Review Letters*, vol. 111, sep 2013. 17, 88

[36] V. I. Sobolev, "Orthogonal projector." Encyclopedia of Mathematics. URL: http://encyclopediaofmath.org/index.php?title=Orthogonal_projector&oldid=48078. 18

[37] N. de Bruijn, "Algebraic theory of penroses non-periodic tilings of the plane. i," *Indagationes Mathematicae (Proceedings)*, vol. 84, no. 1, pp. 39–52, 1981. 20

[38] G. G. Naumis and J. L. Aragón, "Analytic expressions for the vertex coordinates of quasiperiodic lattices," *Zeitschrift für Kristallographie - Crystalline Materials*, vol. 218, jan 2003. 20, 23

[39] B. Boots, A. Okabe, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagram*. Wiley John & Sons, 2000. 32

[40] A. Aho, J. Ullman, and J. Hopcroft, *Data Structures and Algorithms*. Pearson, 1983. 37, 59

[41] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Proceedings of the USSR Academy of Sciencies*, vol. 145, no. 293-294, pp. 595–596, 1963. 39

[42] M. I. S. Franco P. Preparata, *Computational Geometry*. Springer New York, 1993. 42

[43] J. ORourke, C.-B. Chien, T. Olson, and D. Naddor, "A new linear algorithm for intersecting convex polygons," *Computer Graphics and Image Processing*, vol. 19, pp. 384–391, aug 1982. 42

[44] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3. Pearson Education (US), 1998. 45

[45] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, nov 1987. 53, 59

[46] K. Sugihara, "Voronoi diagrams in a river," *International Journal of Computational Geometry & Applications*, vol. 02, pp. 29–48, mar 1992. 54

[47] A. S. Kraemer, M. Schmiedeberg, and D. P. Sanders, "Horizons and free-path distributions in quasiperiodic lorentz gases," *Physical Review E*, vol. 92, nov 2015. 88

[48] J. Marklof and A. Strömbergsson, "The boltzmann-grad limit of the periodic lorentz gas," *Annals of Mathematics*, vol. 174, pp. 225–298, jul 2011. 88

[49] D. Chandler, *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987. 88, 89

[50] S. Torquato, "Hyperuniform states of matter," *Physics Reports*, vol. 745, pp. 1–95, jun 2018. 88