UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE FÍSICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Aplicación de herramientas de negociación automática al problema del ajuste del Voronoi inverso

Supervisor/es: David Orden Martín

José María de la Sen Molina

Grado en Ingeniería de Computadores

Curso académico 2019-20

Convocatoria Septiembre

Aplicación de herramientas de negociación automática al problema del ajuste del Voronoi inverso

Resumen:

Este trabajo de investigación basa su acción en el estudio del guiado eficiente del ajuste del Voronoi inverso, es decir, partiendo de una teselación en el plano se busca colocar un nodo en cada región de manera que las regiones de influencia de estos nodos se ajusten lo mejor posible a la teselación de Voronoi inicial. Se trata de un problema NP-duro y para el tratamiento del mismo se ha empleado una herramienta heurística conocida como "simulated annealing" o recocido simulado, modulada con la capacidad de aceptar configuraciones de nodos que empeoren el resultado con el objetivo de escapar del estancamiento de mínimos y máximos locales. Se ha tratado también de buscar la innovación y aplicar técnicas de negociación entre nodos con el objetivo de mejorar los resultados de las ejecuciones del clásico recocido simulado.

Palabras clave: Voronoi, teselación, recocido simulado, técnicas de negociación

Abstract:

This research work bases its action on the study of efficient guiding of the adjustment of the reverse Voronoi, that is, starting from a tessellation in the plane, it is sought to place a node so that the regions of influence of these nodes adjust as best as possible to the initial Voronoi tessellation. Its an NP-hard problem and an heuristic tool known as simulated annealing has been used to treat it, modulated with the ability to accept node configurations that worse the result in order to escape of local minimums and maximums. An attempt has also been made to seek innovation and apply negotiation techniques between nodes in order to improve the results of executions of the classic simulated annealing.

Keywords: Voronoi, tessellation, simulated annealing, negotiation techniques

Índice

Índice de figuras

Índice de cuadros

1. Introducción

1.1. Resumen extendido

Los **diagramas de Voronoi** aparecen en todo tipo de procesos de crecimiento de la naturaleza, tienen aplicaciones prácticas que van desde la zoología hasta la cristalografía pasando por la silvicultura.

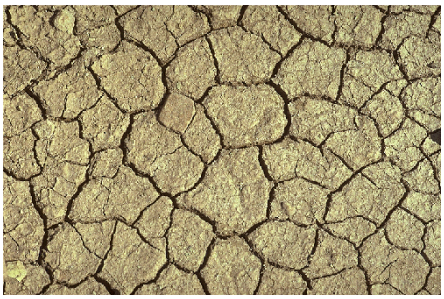

Figura 1: Teselación formada en la tierra

Figura 2: Teselación en la piel de una jirafa

Son una descomposición de un espacio geométrico en regiones, asociada a la presencia de objetos o generadores, de modo que en dicha descomposición se divide el espacio en tantas regiones como puntos u objetos se tengan, de tal forma que a cada punto se le asigne la región formada por todo lo que está más cerca de él que nadie. Son estructuras fundamentales dentro de la Geometría Computacional.

El punto de partida principal y habitual es un conjunto finito de generadores o puntos en el plano $S = \{s_1, s_2, ..., s_n\}$ y una función de distancia euclídea de modo que a cada s_j se le asocian aquellos puntos del plano que están más cerca o igual de s_j que de cualquier otro s_i con i distinto de j. Todo punto del plano quedará entonces dentro de una región. Existirán puntos en el plano que distan lo mismo de dos elementos del conjunto P, conformando fronteras de las regiones anteriormente mencionadas. Se denomina a esta teselación del espacio Diagrama de Voronoi, denotado por $Vor(S)$.

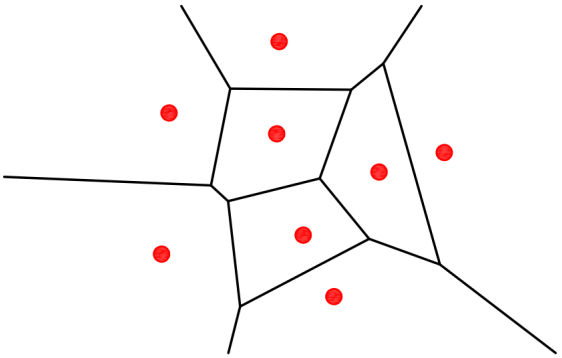

Figura 3: Diagrama de Voronoi básico no delimitado

Este trabajo se plantea desde el punto de vista **inverso**, dada una partición del espacio delimitada por el área de un polígono R se tratará de encontrar el conjunto de generadores cuyo diagrama de Voronoi bajo la función de distancia euclídea se aproxime mejor a la partición dada. La diferencia entre las dos particiones, la original y la aproximada, está definida por el área de su **diferencia simétrica**, es decir, la suma de áreas de los espacios resultantes de la intersección de un polígono de la partición original con el correspondiente de la aproximada y que no pertenecen a ninguno de los dos polígonos involucrados.

La complejidad del problema del Voronoi Inverso no ha sido estudiada todavía y tampoco existe un algoritmo que lo resuelva en tiempo polinomial por lo que se conjetura como un problema NP-duro [2]. En esta investigación se seguirá una herramienta heurística construida en dos pasos o iteraciones. En primer lugar, como se verá en apartados posteriores, se reducirá el espacio de soluciones mediante el algoritmo **Planting Seeds** y finalmente se aplicará el algoritmo **Simulated Annealing**, así como variaciones del mismo mediante el empleo de **técnicas de negociación**, para intentar encontrar un conjunto de soluciones mejor.

2. Base teórica

2.1. Introducción

El objetivo de este Trabajo de Fin de Grado (o TFG) es la construcción de una herramienta heurística que, dado un input en forma de partición en el plano, sea capaz de generar una partición de Voronoi optimizando la diferencia simétrica entre la partición calculada y la original. En consecuencia este TFG se divide en dos grandes fases, una primera de estudio e investigación para el correcto entendimiento de la teoría (estructuras de datos, algoritmos implicados, operaciones geométricas, etc) y una posterior destinada a la experimentación una vez asentadas las bases del proyecto y construidos los flujos principales. En este capítulo se desarrollará toda la teoría necesaria para la subsiguiente experimentación.

2.2. Metaheurística

Se denomina heurística a cualquier aproximación utilizada para resolver un problema a través de un método práctico que no tiene porque ser óptimo, perfecto o racional pero es suficiente para alcanzar una meta a corto plazo. Proviene de la palabra griega *heuriskein* que significa encontrar. Cuando encontrar una solución óptima es imposible, los métodos heurísticos permiten acelerar el proceso para encontrar una solución satisfactoria. En optimización matemática, una metaheurística es un procedimiento de alto nivel (*meta* o más allá) o heurística diseñado para encontrar, generar un algoritmo de búsqueda parcial que pueda proveer una buena solución a un problema de optimización. Las metaheurísticas son estrategias que guían de forma eficiente la exploración y explotación de espacio de soluciones muy grandes permitiendo encontrar soluciones casi óptimas, hacen pocas suposiciones sobre el problema que está siendo resuelto por lo que resultan muy útiles en un amplio abánico de situaciones.

2.2.1. Clasificación de metaheurísticas

Existe un gran número de metaheurísticas y de características a estudiar:

Búsqueda local vs Búsqueda global Este enfoque caracteriza el tipo de estrategia de búsqueda, un ejemplo de algoritmo de búsqueda local es *hill climbing* que trata de buscar óptimos locales pero no garantiza encontrar soluciones óptimas globales, una aplicación del mismo se puede ver en [3] dónde se desarrolla el problema de las n-reinas y una posible solución empleando la heurística de *hill climbing* para resolverlo de forma eficiente, utilizando como función objetivo $O(tablero) = (N^2/2) - L$ donde L denota el nº de parejas de reinas atacándose en el tablero actual y $N^2/2$ el nº de parejas de reinas posibles, una solución consistiría en encontrar un tablero con el máximo valor de la función objetivo, por ejemplo, $N^2/2$.

Otras en cambio tratan de buscar óptimos locales con el objetivo de encontrar óptimos globales, algunas de estas estrategias son *simulated annealing*, *tabu search* o *GRASP*. Esta última metaheurística, *Greedy Randomized Adaptive Search Procedure*, modela un proceso iterativo o de múltiples entradas en el que cada iteración basa su acción en dos fases: una de construcción y una de búsqueda local. La fase de construcción modela una solución factible cuya vecindad será investigada hasta que la fase de búsqueda local encuentre un mínimo local [4].

Finalmente se señalan estrategias que basan su acción estrictamente en la globalidad como pueden ser: *ACO (ant colony optimization)* en el que se persigue la mejor solución global a través de la analogía de las feromonas de las hormigas en cada iteración [5], *PSO (particle swarm optimization)* en el que para encontrar la solución óptima cada partícula de la población se mueve en la dirección su anterior mejor solución y a la vez, en dirección de la mejor solución global en el enjambre [6] o la *computación evolutiva* que trata de imitar los procesos evolutivos (cruzamiento, mutación...) permitiendo la superviviencia de los mejores individuos para construir modelos de búsqueda y optimización.

Solución única vs Solución basada en población Se diferencian dos tipos de metaheurísticas, aquellas que se centran en modificar y mejorar una única solución candidata como es *simulated annealing* y aquellas que trabajan con múltiples soluciones (población) como es la optimización por *enjambre de partículas*, en la que cada solución candidata o partícula se mueve a la vez por el espacio de búsqueda de forma individual y de forma colectiva.

Algoritmos de hibridación vs Algoritmos meméticos Los primeros combinan una metaheurística con otras técnicas de optimización mediante la compartición de información como por ejemplo el *aprendizaje automático*, los segundos representan la sinergia de la evolución de cualquier población y el propio aprendizaje de los individuos de la población (procedimientos de mejora local). Un ejemplo consistiría en emplear búsqueda local en lugar del *operador mutación* en los *algoritmos evolutivos* [7].

Basadas en la naturaleza y en métaforas Un área de investigación interesante es el diseño de metaheurísticas basadas en sistemas y mecanismos propios de la naturaleza, entre las que se destaca *simulated annealing* que se verá en secciones futuras, basado en el recocido de los metales, *ACO* basada en el movimiento de las hormigas dentro de las colonias, originado por la comunicación indirecta entre ellas a través del rastro de feromonas químicas que dejan, lo que les permite encontrar los caminos más cortos entre su nido y las fuentes de alimentos [5], o *PSO* que imita el comportamiento social de los bancos de peces en el mar, de los pájaros en el cielo, etc.

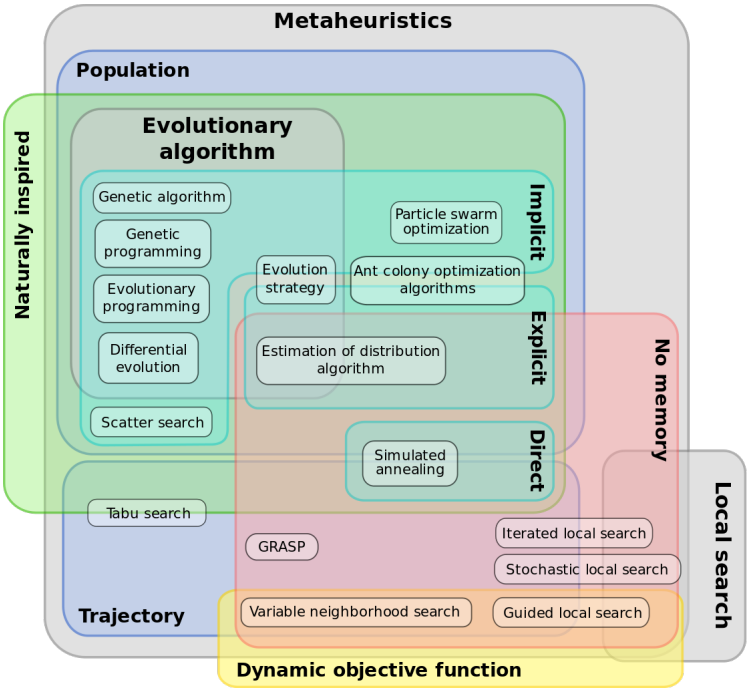

Figura 4: Clasificación de las metaheurísticas [1]

2.3. Diagrama de Voronoi

En primer lugar se debe señalar que para cada entero positivo d existirán diagramas de Voronoi de d dimensiones, en este proyecto solo se trabajará con teselaciones de dos dimensiones.

Un diagrama de Voronoi de un conjunto de sitios o puntos generadores es una colección de regiones que dividen el plano. Cada región se corresponde con uno de los generadores y todos los puntos en el interior de una región están más cerca del generador de esa región que de cualquiera de los otros generadores. Existirán puntos en el espacio que equidistarán de dos generadores y formarán un límite o borde compartido entre dos regiones. Por ejemplo, en la figura 5, se puede apreciar que el punto s está más cerca del generador s_1 que de cualquiera de los otros generadores, también se puede observar que el punto s', que se encuentra en un borde, equidista de de s_1 y s_3.

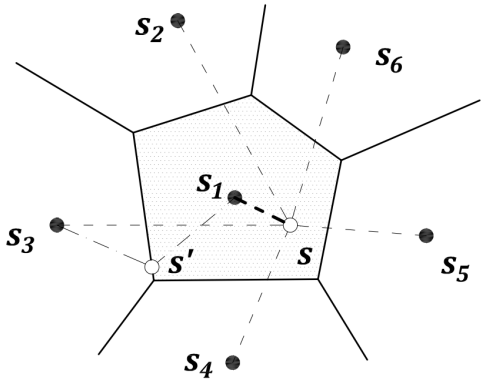

Figura 5: Diagrama de Voronoi con seis generadores

Las aplicaciones del Diagrama de Voronoi son múltiples tanto dentro como fuera del mundo matemático, tanto como métodos para resolver problemas como para definir modelos ya existentes. Son muy útiles en geometría computacional para definir problemas de cuantización, en robótica para definir problemas de detección de obstáculos, para modelar occurrencias de la naturaleza como por ejemplo la competición entre especies de plantas (ecología y silvicultura), territorios de animales (zoología) y de clanes y tribus (antropología y arqueología), y patrones de asentamientos urbanos (geografía) [8].

2.3.1. Definición formal del diagrama de Voronoi

Se define formalmente el diagrama de Voronoi ordinario de dos dimensiones; en primer lugar se denota la localización de un punto s_i como (s_{i1}, s_{i2}) y el correspondiente vector como $\overrightarrow{s}$. Se establece $S = \{s_1, s_2, ..., s_n\} \in \mathbb{R}^2$, donde $2 \leqslant n < \infty$ e $s_i \neq s_j$, $i \neq j$ y $\forall i, j = 1, 2, ..., n$ como el conjunto de puntos generadores. Se define una región de Voronoi por: $VR(s_i, S) = V_i = \{\overrightarrow{s} \quad | \quad \|\overrightarrow{s} - \overrightarrow{s_i}\| \leqslant \|\overrightarrow{s} - \overrightarrow{s_j}\| \quad \forall j \ni i \neq j\}$ dónde $\|\cdot\|$ denota la distancia euclidea. Todas las regiones en un diagrama de Voronoi ordinario están conectadas y son convexas, se puede definir este conjunto de regiones como: $VD(S) = VD = \{V_1, V_2, ..., V_n\}$, el diagrama de Voronoi de S. Otra notación es: $VD(S) = VD = \bigcup_{i=1}^{n} V_i$.

2.3.2. Componentes básicos del diagrama de Voronoi

Está compuesto de tres elementos: los generadores, las aristas y los vértices.

Generadores El conjunto S visto anteriormente.

Aristas Una arista entre las regiones de Voronoi V_i y V_j es $V_i \bigcap V_j = e(s_i, s_j)$. Si $e(s_i, s_j) \neq \emptyset$, V_i y V_j son adyacentes. El conjunto de aristas formando una región de Voronoi V_i puede ser denotado por $\partial(VR(s_i)) = \partial(V_i)$.

Vértices Un vértice es cualquier punto que equidiste de tres o más puntos generadores en el plano. Se denotan como q_i y conforman las fronteras de las aristas. El número de aristas que desembocan en un vértice se denomina grado del vértice. Si $degree(q_i) = 3 \ \forall q_i$, VD es de tipo no-degenerado, en caso contrario, VD se considera degenerado.

2.3.3. Diagramas de Voronoi en subconjunto acotado de $\mathbb{R}^2$

La mayor parte de las veces se consideran los diagramas de Voronoi en $\mathbb{R}^2$, pero también se puede trabajar con ellos en cualquier subconjunto $BS \subseteq \mathbb{R}^2$. Asumiendo BS como no vacío, un diagrama de Voronoi delimitado se definiría por: $VD_{\cap BS} = \{V_1 \cap BS, V_2 \cap BS, ..., V_n \cap BS\}$. Si para cualquier i, V_i comparte parte del límite o cota de BS, se denomina a $V_1 \cap BS$ región límite. En esta investigación se empleará un subconjunto $BS \subseteq \mathbb{R}^2$ delimitante con disposición rectangular.

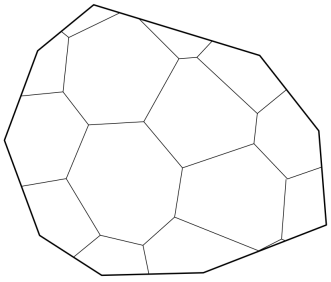

Figura 6: Diagrama de Voronoi en subconjunto acotado de $\mathbb{R}^2$

2.3.4. Regiones de dominio

Dados dos puntos generadores, s_i y s_j, el bisector perpendicular a la línea que conecta s_i y s_j se define como: $b(s_i, s_j) = \{\overrightarrow{s} \quad | \quad \|\overrightarrow{s} - \overrightarrow{s_i}\| = \|\overrightarrow{s} - \overrightarrow{s_j}\|, i \neq j$. La región de dominio de s_i sobre s_j: $H(p_i, p_j) = \{\overrightarrow{s} \quad | \quad \|\overrightarrow{s} - \overrightarrow{s_i}\| \leqslant \|\overrightarrow{s} - \overrightarrow{s_j}\|\}$, $i \neq j$, formada por todos los puntos del plano que están más cerca de s_i que de s_j o que equidistan de ambos, también se denota por $Dom(s_i, s_j)$.

Partiendo de esta definición de las regiones de dominio, se puede definir una región de Voronoi de forma alternativa: $VR(s_i, S) = V_i = \bigcap_{j \in \mathbb{Z}^+ \leqslant n} H(p_i, p_j)$, de modo que el conjunto $VD(S) = VD = \{V_1, V_2, ..., V_n\}$ es el diagrama de Voronoi en $\mathbb{R}^2$ generado por S.

En la figura 7 se puede observar la región de dominio s_1 sobre s_2 con el patrón hexagonal amarillo, la de s_1 sobre s_3 en línea rosas, la de s_1 sobre s_4 en virutas verdes, la de s_2 sobre s_1 con flechas moradas, la de s_3 sobre s_1 con una cuadrícula roja y finalmente, la de s_4 sobre s_1 con signos positivos azules.

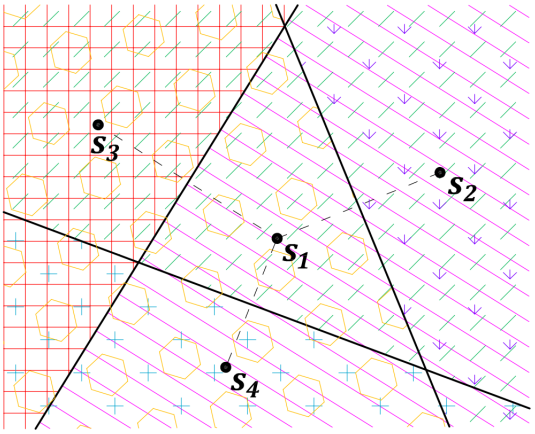

Figura 7: Regiones de dominio formadas por cuatro generadores

2.3.5. Envolvente convexa

Es el menor de los subconjuntos convexos de $\mathbb{R}^2$ que contiene el conjunto de generadores S, se denota como $CH(S)$. La envolvente es referida como $\partial(CH(S))$. Dado un diagrama de Voronoi $VD(S)$, la región V_i no está delimitada si y sólo si $s_i \in \partial(CH(S))$, además teniendo en cuenta el concepto de $CH(S)$ se puede establecer:

i Una arista de Voronoi $e(s_i, s_j)(\neq \emptyset)$ es un segmento de línea si y sólo si la línea conectando s_i y s_j ($\overline{s_i s_j}$) no está en $\partial(CH(S))$.

ii Una arista de Voronoi $e(s_i, s_j)(\neq \emptyset)$ es un semi-segmento si S no es colineal y s_i y s_j son generadores consecutivos en $\partial(CH(S))$.

iii Todos las aristas del diagrama de Voronoi son rectas si S es colineal.

2.3.6. Círculos vacíos

Para cada vértice $q_i \in VD(S)$ existe un círculo vacío C_i centrado en q_i que pasa por al menos tres puntos generadores.

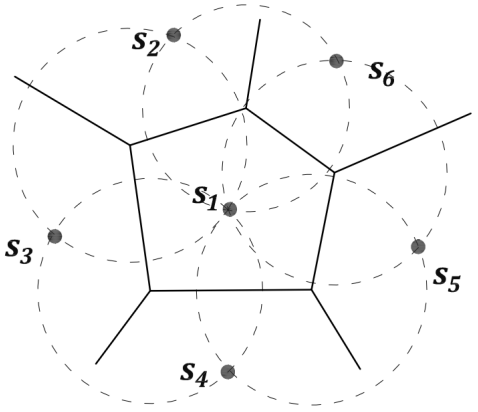

Figura 8: Círculos vacíos formados por una región de Voronoi de cinco vértices

2.4. Diferencia simétrica

En todas las técnicas de optimización se emplea una medida de bondad de la solución a medida que se explora el espacio de soluciones, en el caso de este proyecto se empleará la

diferencia simétrica.

Dada una partición P con n regiones se asigna a cada región un índice i. Dado un conjunto de n sitios S se asocia cada sitio $s_i \in S$ a la región de la partición cuyo índice es i y se emplea $P(s_i)$ para la región en P asociada al sitio s_i. Se emplea $VD(S)$ para definir al diagrama de Voronoi del conjunto S y $VR(s_i, S)$ para definir a la región del diagrama de Voronoi con sitio $s_i \in S$. Dados un conjunto S y una partición P, se emplea $sd_p(s_i, S)$ para el área de diferencia simétrica entre $P(s_i)$ y $VR(s_i, S)$, es decir, el área de diferencia no común resultante de la intersección entre los polígonos formados por $P(s_i)$ y $VR(s_i, S)$ (figura 9) [2]. Se define la diferencia simétrica del sistema como:

$$SD_p(S) = \frac{\sum_{i\in[1,n]} sd_p(s_i, S),}{2}$$

Es necesario destacar que el cálculo local $sd_p(s_i, S)$ siempre queda normalizado respecto al área rectángular por la que queda delimitado el Voronoi y con ello el global, para permitir el posterior contraste con el artículo [2].

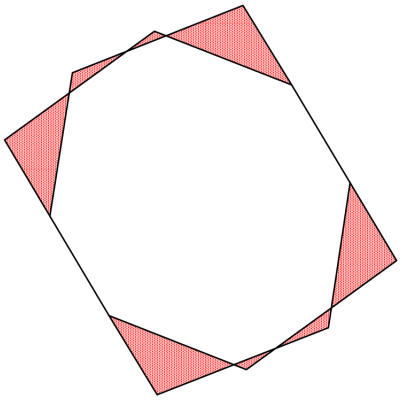

Figura 9: Diferencia simétrica en rojo entre dos polígonos

Para posteriormente comprender la construcción de la herramienta heurística empleando como base el algoritmo recocido simulado (simulated annealing) se deben tener claros los siguientes conceptos:

Solución Conjunto S de n sitios en el plano.

Solución vecina Conjunto S con un punto solución alterado o desplazado.

Coste de la solución La diferencia simétrica entre el diagrama de Voronoi y la partición de entrada, $SD_p(S)$, es la función a optimizar.

Solución inicial Conjunto generado tras la ejecución del algoritmo que se verá en 2.6.1, *Planting Seeds*.

2.5. Modelado estocástico

El modelado estocástico es un área interesante y desafiante de la probabilidad y la estadística. Un **proceso estocástico** es una **familia de variables aleatorias** X_θ, indexadas por un parámetro θ, dónde θ pertenece a un conjunto de índices Θ. Estos modelos pueden ser contrastados con los modelos deterministas, conjunto de ecuaciones que describen de forma exacta como evolucionará el sistema a lo largo de Θ. En un modelo estocástico la evolución es parcialmente aleatoria y aunque se ejecute infinidad de veces el sistema no dará resultados idénticos. Las diferentes ejecuciones se denominan **realizaciones del proceso**. En la mayoría de los ejemplos existentes Θ representa el tiempo, en el caso de esta investigación representará la temperatura del *annealing*, es decir, n valores discretos y conocidos. Además la familia de variables aleatorias vendrá dada por el conjunto de generadores que garanticen la mejor solución en la temperatura iterada, de modo que se puede denotar el proceso estocástico como: S_n. En un proceso discreto, la variable S_n dependerá de los valores anteriores del proceso $S_{n-1}, S_{n_2}, ...$, de modo que se puede definir la distribución condicional como:

$$Pr(S_{nk}|S_{n_{k-1}}, S_{n_{k-2}}, ..., S_{n_1})$$

Se debe prestar especial atención en el cumplimiento de la **propiedad de Markov** por parte del proceso estocástico que se estudia, que establece que:

$$Pr(S_{nk}|S_{n_{k-1}}, S_{n_{k-2}}, ..., S_{n_1}) = Pr(S_{nk}|S_{n_{k-1}})$$

Esta propiedad es denominada de este modo por el matemático ruso Andréi Márkov (1856-1922), en ocasiones también es referida como la propiedad de la "falta de memoria". Un mnemónico para recordar esta propiedad es: "Dado el presente (S_{k-1}), el futuro (S_k) es independiente del pasado $(S_{k-1}, S_{k-2}, ..., S_{n_1})$"

2.6. Algoritmos

En este apartado se desarrollarán los algoritmos principales utilizados.

2.6.1. Planting Seeds

Como bien se ha señalado en el desarrollo del concepto diferencia simétrica en esta problemática se parte de una partición del espacio P, con el objetivo de iniciar la resolución se debe generar un conjunto inicial de soluciones, compuesto por semillas o puntos en el espacio contenidos cada uno en una región de la partición inicial. Se recuerda que dos celdas vecinas de un diagrama de Voronoi contienen cada una un punto generador simétrico respecto a la arista común de las celdas, teniendo esto en cuenta se establece como principio básico para la construcción del algoritmo Planting Seeds:

Se define $S = \{s_1, ..., s_n\}$ como un conjunto de n sitios o puntos en el plano. Para cada celda de Voronoi $VR(s_i, S)$ se define G_i como la intersección de sus polígonos vecinos reflejados sobre la arista común. Se genera un conjunto S que verifica $s_i \in G_i$ para cada $i \in [1, n]$.

De forma más detallada, teniendo en cuenta la definición de G_i y siendo $G_i^o = G_i$ para cada $i \in [1, n]$. Para cada $j > 0$ se define G_i^j como la intersección de G_i^{j-1} y las regiones poligonales obtenidas de reflejar las regiones poligonales G_k^{j-1} para todo k que cumpla que $VR(s_k, S)$ es una región vecina de $VR(s_i, S)$ respecto a la arista común. Iterando con este método para $j = 0, 1, ...$ se obtiene para cada celda de Voronoi $VR(s_i, S)$ una secuencia de regiones convexas anidadas tal que $VR(s_i, S) \supseteq G_i^0 \supseteq G_i^1 \supseteq G_i^2 ...$ todas ellas conteniendo la semilla s_i que genera V_i. Generalmente el límite de esta secuencia es un círculo con centro s_i en la región $VR(s_i, S)$ cuyo diámetro es la distancia entre los dos sitios más cercanos del diagrama de Voronoi. Estas regiones definidas nos permiten calcular un generador en cada G_i satisfaciendo ciertas propiedades simétricas. Ya que en ocasiones se tendrá como entrada divisiones del espacio que no son teselaciones de Voronoi, en alguna iteración j este procedimiento generará una región vacía G_i^j por lo que se empleará la iteración $j - 1$ y el baricentro de la región generada como semilla inicial del vecino i [2]. Empleando este algoritmo, con las regiones convexas generadas, se limita la futura exploración y explotación del espacio de soluciones. Se puede observar esta limitación en las figuras 10 y 11.

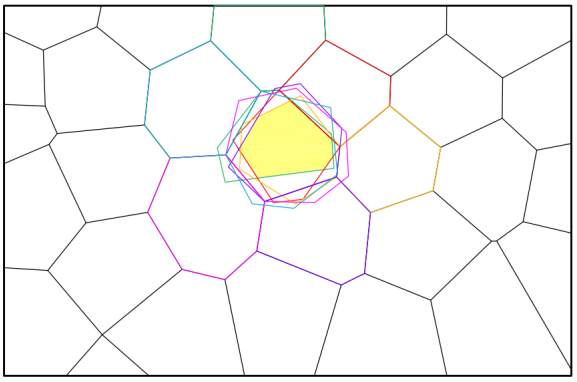

Figura 10: Reflejo de los polígonos tras la primera iteración del algoritmo Planting Seeds

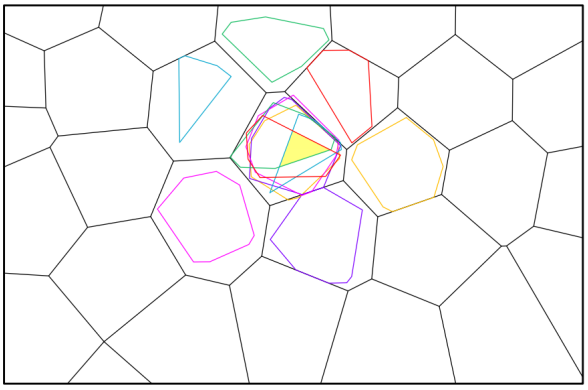

Figura 11: Reflejo de los polígonos tras la segunda iteración del algoritmo Planting Seeds

2.6.2. Recocido Simulado

Finalmente se emplea el algoritmo Recocido Simulado o *Simulated Annealing (SA)* que mediante un buen conjunto de semillas S y unos parámetros de entrada correctos permite encontrar soluciones casi óptimas. En este trabajo se emplean los parámetros de entrada del artículo [2], escogidos tras realizar cientos de experimentos satisfactorios para particiones del espacio con menos de 50 generadores cercanas a ser consideradas diagramas de Voronoi.

El algoritmo Recocido Simulado es uno de los métodos heurísticos preferidos para resolver problemas de optimización. Kirkpatrick introdujo SA inspirándose en el proceso de recocido del metal. Este proceso define la disposición molecular óptima dónde la energía potencial de la masa es minimizada al enfriar los metales tras someterse a altas temperaturas. De manera general el algoritmo SA adopta un movimiento iterativo acorde al parámetro de temperatura

variable que imita la transición de recocido de los metales. Una implementación básica del algoritmo consiste en comparar iterativamente los resultados de las funciones objetivo a optimizar con entradas actuales y con entradas vecinas o alteradas de modo que si la nueva solución mejora la salida de las funciones objetivo es guardada como solución base para la siguiente iteración, de no proceder de esta forma el algoritmo finalizaría el procedimiento sin buscar en un dominio más amplio, sin obtener mejores resultados debido a estar atrapado en mínimos o máximos locales. El algoritmo SA propone una solución efectiva a la problemática anterior mediante la incorporación de dos bucles iterativos: uno que permite controlar el **enfriamiento del proceso de recocido** y otro aplicando el **criterio de Metrópolis** [9].

Metal	**Recocido Simulado**
Metal	Problema
Sistema de estados	Soluciones factibles
Nivel de energía	Función de coste
Cambio de estado	Solución vecina
Temperatura	Parámetro de control
Ordenamiento completo de la estructura cristalina	Solución óptima del problema

Cuadro 1: Analogía entre el metal y el recocido simulado

Criterio de Metrópolis La idea básica tras este criterio es explorar el espacio de soluciones vecino a la solución candidata con el objetivo de evitar mínimos y máximos locales permitiendo movimientos "no recomendados". Como ejemplo básico de problema optimización se define una función objetivo $f(x_i)$ siendo el conjunto $x_i = \{x_1, x_2, ..., x_n\}$ con $n \in R$. De este modo si $f(x_{i+1}) < f(x_i)$ se toma x_{i+1} como nueva solución candidata para la siguiente iteración del bucle, en caso contrario se emplea una función de aceptación $w = exp[\frac{-f(x_{i+1})-f(x_i)}{T_c}]$ donde T_c es la temperatura actual y se genera un número aleatorio s que cumpla $0 < s < 1$. Finalmente si $w > s$ se acepta x_{i+1}, en caso contrario se rechaza la solución vecina.

Enfriamiento del proceso de recocido Este proceso se lleva a cabo mediante el control y descenso de la variable de temperatura del algoritmo. El algoritmo Metrópolis como se ha visto propone una solución con una temperatura constante. Para valores altos de T_c el algoritmo requiere un espacio de búsqueda más amplio acorde con el movimiento de las partículas del metal a temperaturas altas, de este modo la probabilidad para alcanzar el mínimo global aumenta, de forma contraria para valores pequeños de T_c se requiere un espacio de búsqueda más pequeño y el riesgo de quedar atrapado en un mínimo local aumenta acorde con la estabilidad de los metales a temperaturas bajas.

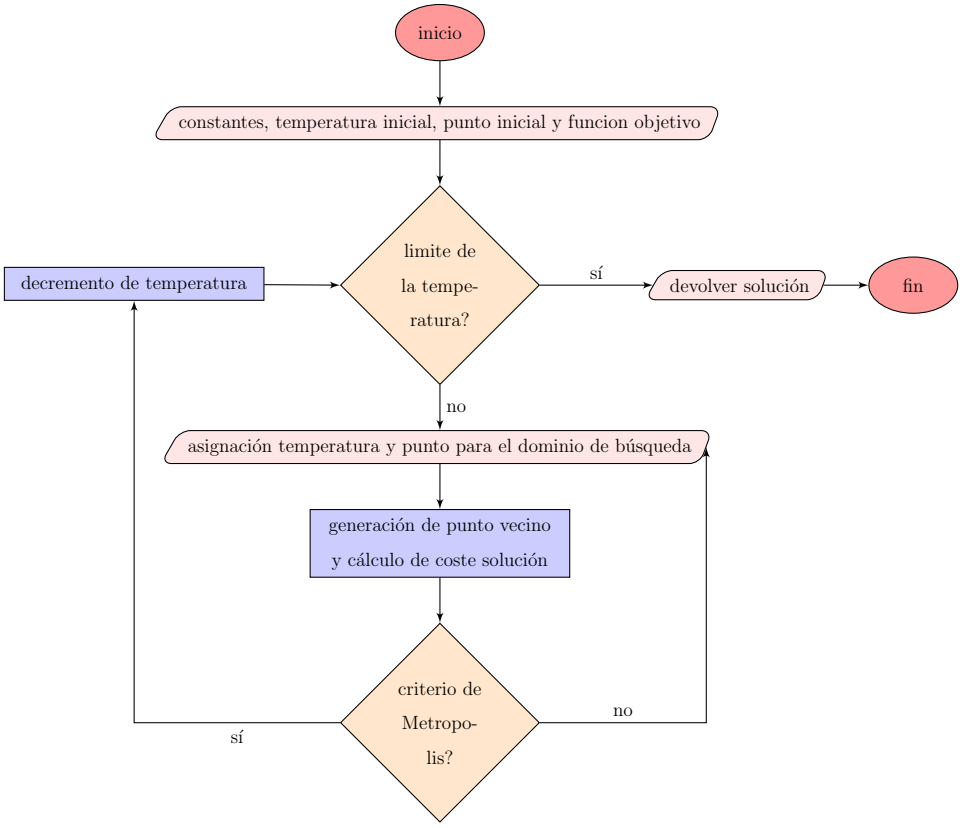

Figura 12: Flujo SA

Especificación de parámetros para SA Como se ha señalado anteriormente el espacio de soluciones del problema a tratar, el ajuste del Voronoi inverso, está compuesto por $S = s_1, ..., s_n$, conjunto de n sitios o puntos en el plano, donde n es el número de particiones de la región del espacio que sirve como entrada. En esta investigación se emplean los parámetros de entrada del artículo [2] obtenidos tras la realización de un alto número de experimentos,

estos resultan eficientes y satisfactorios para particiones entrada con menos de 50 sitios, la partición de entrada de este experimento estará compuesta de $n = 29$ sitios. Se establecen los siguientes parámetros:

Temperatura inicial $t_0 = \frac{1}{log_{10}(SD_p(S_0))}$

Radio de enfriamiento $r = 0{,}7 + log_{10}(n)$

Nº iteraciones por temperatura $L = t/(l * n)$ dónde l es 0,06 o el valor que se introduzca por parte del investigador.

Aceptación de la solución $w = e^{-\delta/t}$, dónde δ es la diferencia de energía entre la nueva solución candidata y la solución actual, la diferencia simétrica.

Límite de la temperatura $t_{fin} = t_0/100log_{10}(n)$

Como se puede apreciar el valor n afecta a todos los parámetros anteriores, esto se debe a que cuánto más regiones tenga una partición más locales son los cambios en la solución.

2.6.3. Técnicas de negociación

Con el objetivo de mejorar los resultados obtenidos con el algoritmo base simulated annealing se desarrollarán una serie de técnicas de negociación entre nodos mediante la construcción de dos tipos de políticas: la **relación entre nodos** y la **técnica de aceptación**. Se tratará de explotar las relaciones de localidad por encima de la de globalidad modificando el criterio de Metrópolis.

Relación entre nodos Se refiere a la elección de parejas de nodos, de comunidades de nodos (un nodo y todos los nodos vecinos), a nodos con un color común (con una previa coloración aleatoria o siguiendo una división geométrica del espacio)... cuando se calcula la energía y la aceptación de la solución.

Parejas de nodos Del conjunto S' se estudia la energía local del punto alterado s_i y uno de sus vecinos (punto generador perteneciente a una de las caras adyacentes a la cara de s_i). Véase la figura 13, en verde la cara de la iteración, en morado el vecino escogido y en amarillo los vecinos que podrían haber sido escogidos.

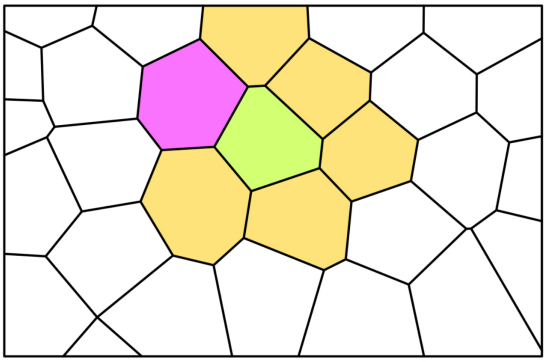

Figura 13: Relación por parejas de nodos

Comunidad de nodos por vecindad Del conjunto S' se estudia la energía local del punto alterado s_i y la de cada uno de sus vecinos, aquellos generadores pertenecientes a las regiones adyacentes. Véase la figura 14, en verde la cara de la iteración y en morado los vecinos.

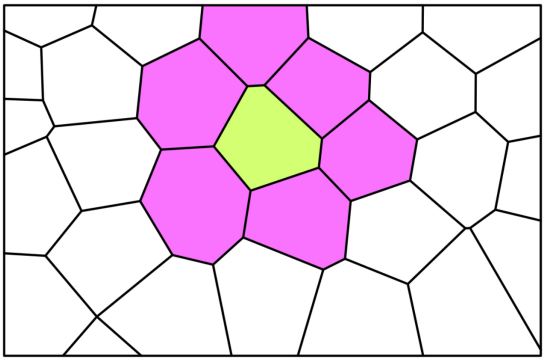

Figura 14: Relación por comunidad de vecinos

Comunidad de nodos por coloración Del conjunto S' se estudia la energía local del punto alterado s_i y la de cada uno de los generadores homólogos de color. Véase la figura 15.

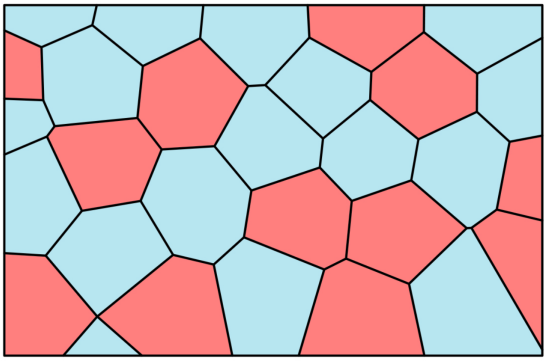

Figura 15: Relación por comunidad de colores

A partir de ahora se hará referencia a los **subconjuntos** anteriores como S'' y a la cardinalidad $|S''|$ como n''.

Técnica de aceptación Se refiere al criterio de evaluación seguido cuando la energía y la aceptancia de la solución dependen de más de un nodo, es decir, se hará filtrado mediante cadenas de conjunción u operaciones de tipo "and", cadenas de disyunción u operaciones de tipo "or", cantidad de nodos mejorados, etc. La estructura es de estas técnicas es siempre la misma compuesta de dos "subcriterios":

Cadena de conjunción En el caso de que la **energía local** de **todos** los nodos mejore se acepta la nueva solución, en caso contrario se sigue el criterio de Metrópolis estableciendo:

$$0 < s < 1$$

$$w = e^{-\delta/t}$$

$$\delta = SD_{p-and}(S'') = \frac{\sum sd_p(s_i, S''), i \in [1, n'']}{2}$$

Cadena de disyunción En el caso de que la **energía local** de **uno** de los nodos mejore se acepta la nueva solución, en caso contrario se sigue el criterio de Metrópolis estableciendo:

$$0 < s < 1$$

$$w = e^{-\delta/t}$$

$$\delta = SD_{p-or}(S'') = \frac{\sum sd_p(s_i, S''), i \in [1, n'']}{2}$$

Nº de nodos mejorados En el caso de que la **energía local** de al menos la **mitad** los nodos mejore se acepta la nueva solución, en caso contrario se sigue el criterio de Metrópolis estableciendo:

$$0 < s < 1$$

$$w = e^{-\delta/t}$$

$$\delta = SD_{p-groups}(S'') = \frac{\sum sd_p(s_i, S''), i \in [1, n'']}{2}$$

Esta es la idea base de las relaciones y las técnicas, posteriormente se realizarán modificaciones sobre estos planteamientos con el objetivo de mejorar los resultados de las políticas y tratar de aproximarse a la perspectiva clásica del algoritmo.

El flujo de la combinación de la construcción del Voronoi, con simulated annealing y con las técnicas de negociación sería:

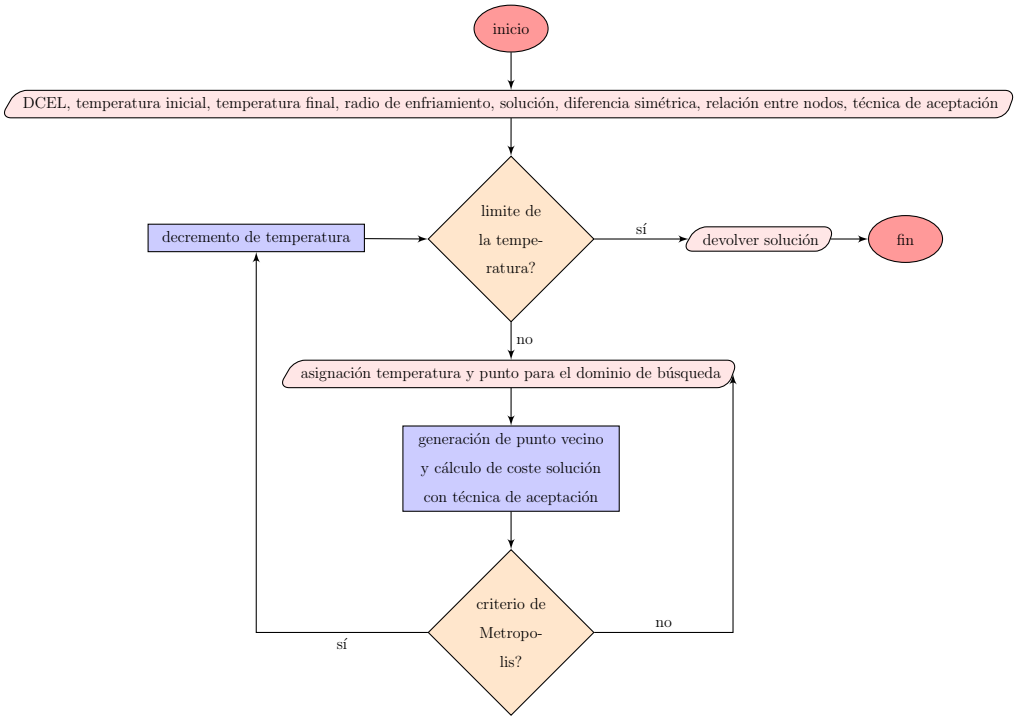

Figura 16: SA (Voronoi inverso) + técnica de aceptación

2.6.4. Perspectiva general

Una vez explicados los algoritmos principales junto con las técnicas de negociación es momento de visualizar la foto global de la herramienta heurística construida, se recuerda que se reducirá el espacio de soluciones con el algoritmo Planting Seeds y se tratará de guiar de forma eficiente el proceso estocástico de la optimización del Recocido Simulado empleando técnicas de negociación entre los nodos implicados:

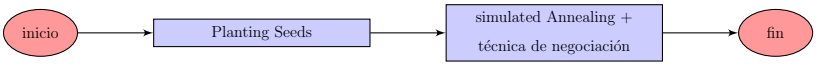

Figura 17: Flujo global de la herramienta heurística construida

2.7. Modelado de datos

Muchos algoritmos del campo de la geometría computacional trabajan con grafos planos, la lista de aristas doblemente enlazada o DCEL (doubly connected edge list) es una estructura de datos que permite representar, manipular y recorrer este tipo de grafos. Un grafo planar es aquel que se puede dibujar en el plano sin que sus aristas se crucen. Se puede incluir el esquema de vértices y segmentos entre vértices que componen el diagrama de Voronoi dentro de esta clasificación.

2.7.1. DCEL

El componente principal del DCEL es una lista de medias-aristas. Cada arista en el DCEL está representada por dos medias-aristas, opuestas en dirección, denominadas gemelas. Las medias-aristas dentro de una cara están orientadas en sentido opuesto a las agujas del reloj.

Un DCEL almacena una lista de medias-aristas, una lista de vértices y una lista de caras. Estas listas no están ordenadas pero están interconectadas por varios punteros. Aparte de estos elementos se guardan una serie de metadatos para facilitar la construcción de la herramienta basada en el *annealing*:

Vértice Un vértice almacena las coordenadas del propio punto que representa, un puntero a la media-arista de la que es origen y un puntero a la media-arista que finaliza en él.

Artista Una media-arista almacena 5 punteros, uno al vértice objetivo, uno a la cara incidente, uno a la media-arista gemela, uno a la media-arista siguiente de la cara incidente y otro a la media-arista anterior de la cara incidente. Como metadatos se almacena la longitud de la arista y el ángulo que forma con el eje X.

Cara Una cara almacena un puntero a una media-arista dentro de la propia cara. Entre la metainformación salvada para una cara se puede encontrar el polígono resultante de las ejecuciones del algoritmo Planting Seeds, el punto generador que da origen a la región de Voronoi asociada a la cara, el color otorgado a la cara para el posterior *annealing* con negociación, etc.

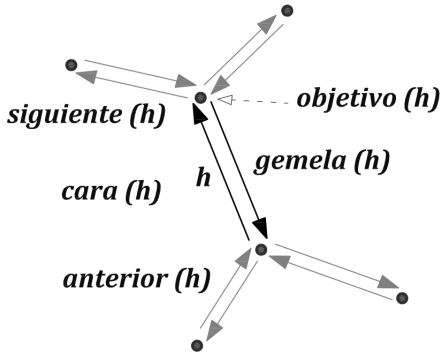

Figura 18: Medias-aristas en DCEL

2.8. Operaciones geométricas

La herramienta heurística se apoya constantemente en una serie de operaciones geométricas a medida que itera entre los diferentes algoritmos, se detallan a continuación:

2.8.1. Ordenación de aristas

Con el objetivo de controlar en un vértice el orden de la lista de aristas en sentido antihorario se emplean las propiedades trigonométricas. Teniendo dos aristas A y B se calcula el módulo (hipotenusa) y el desplazamiento de los vértices no comunes en el eje X (cateto adyacente) de ambas, finalmente se calcula el arcoseno $\arccos(\frac{h}{b})$ para determinar el ángulo formado con el eje de abscisas.

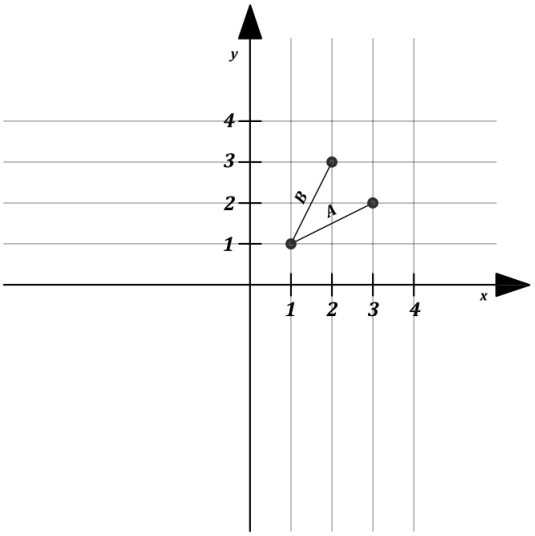

Figura 19: Ordenación de aristas

2.8.2. Determinante de Gauss

Se trata de un procedimiento muy útil para calcular el área de cualquier polígono irregular. Se elige un vértice aleatoriamente del polígono y en sentido antihorario se extraen las coordenadas de los vértices del polígono, finalmente se computa:

$$A = \frac{1}{2}\left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + ... + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix}\right)$$

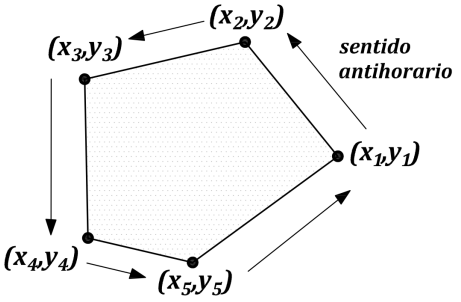

Figura 20: Área del polígono irregular

2.8.3. Cálculo de simétricos

Para el cálculo de caras y polígonos simétricos respecto a las aristas que se realiza en el algoritmo *Planting Seeds*, se utiliza el concepto de punto simétrico respecto a una recta, de

este modo se computan todos los simétricos de la cara vecina deseada. Para el cálculo del simétrico se traza una perpendicular a la recta formada por los vértices de la arista iterada, q_i y q_j, que pase por el vértice a simetrizar de la cara vecina o polígono (tras varias ejecuciones del algoritmo Planting Seeds) q_k.

2.8.4. Excitación de puntos

Para la excitación de los puntos necesaria en las iteraciones del Recocido Simulado se calcula un ángulo de excitación aleatoriamente comprendido entre 0 y 2π, y un módulo de excitación comprendido entre 0 y la distancia entre el punto y el vértice más cercano. De este modo se compone el vector velocidad por el que el punto se verá excitado. Véase la figura 21.

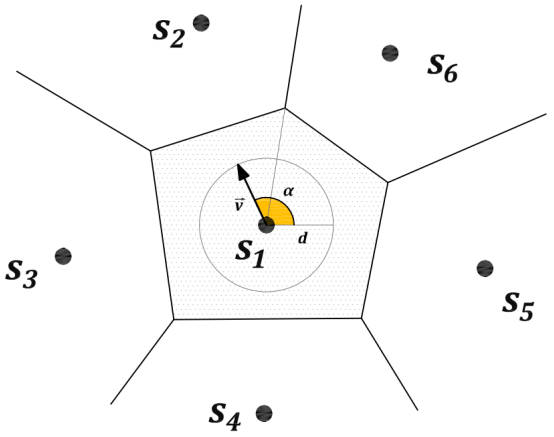

Figura 21: Excitación de punto generador

3. Experimentos

Una vez construida la herramienta es hora de realizar un conjunto de experimentos con las diversas técnicas de negociación y aceptación planteadas. Para ello se realizan los siguientes tipos de ejecuciones:

- Simulated Annealing clásico
- simulated Annealing por parejas
  - Aceptación por conjunción
  - Aceptación por disyunción
- simulated Annealing por comunidad de nodos por vecindad
  - Aceptación por conjunción
  - Aceptación por disyunción
  - Aceptación por nº de nodos mejorados
- simulated Annealing por comunidad de nodos por coloración
  - Aceptación por conjunción
  - Aceptación por disyunción
  - Aceptación por nº de nodos mejorados

Téngase en cuenta que la aceptación por nº de nodos mejorados con comunidades basadas en parejas no tiene sentido, es equivalente a la aceptación por disyunción. Para el testeo de la comunidad de nodos por coloración se han pintado las caras de la partición de entrada de dos formas: la primera, de forma aleatoria con dos colores (azul y rojo), la segunda, formando cuatro rectángulos en el espacio (dos rojos y dos azules). La coloración es estática, es decir, no cambia la disposición de la coloración a lo largo de las ejecuciones de cada experimento. Véanse las figuras 22 y 23. Todos los experimentos sobre el *annealing* parten del mismo número de ejecuciones previas del algoritmo Planting Seeds.

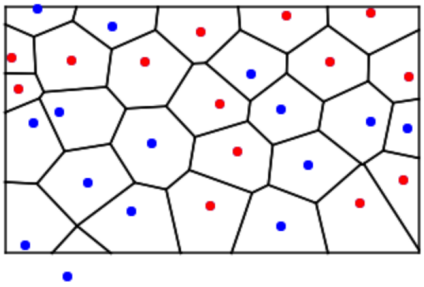

Figura 22: Distribución aleatoria

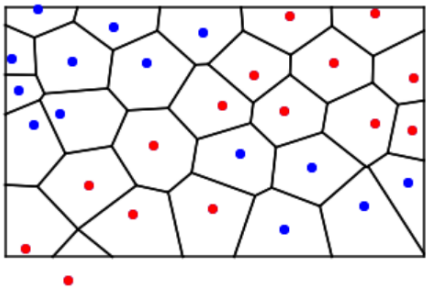

Figura 23: Distribución por rectángulos

4. Resultados

Se muestran una serie de tablas resumiendo las ejecuciones para cada experimento, **PS** denota las ejecuciones realizadas del algoritmo Planting Seeds, **r** el radio de enfriamiento de la temperatura, **l** el parámetro utilizado para calcular el número de iteraciones a computar a una temperatura determinada como en 2.6.2, **result** la diferencia simétrica media de las ejecuciones, **sDev** la desviación típica del experimento, **time** el tiempo medio de las ejecuciones en formato HH:mm:ss, **acc** el porcentaje de aceptados sobre el total, **den** el porcentaje de denegados sobre el total $(1 - acc)$ y **n** el número de ejecuciones realizadas para ese experimento. Se estudia de forma detallada:

4.1. simulated Annealing

4.1.1. Classic

Se trata de la perspectiva sin técnicas de negociación, con ella se han obtenido los mejores resultados medios (tablas 2). En las figuras 24 y 25 se pueden apreciar unas muy buenas evoluciones del proceso estocástico, la evolución de la herramienta con negociación tiene que tratar de asimilarse a estos desarrollos. En esta aproximación se ha escogido una cara aleatoriamente en cada iteración, se ha excitado el punto generador de la misma y se ha estudiado la diferencia simétrica global, en el caso de mejorar la solución es aceptada, en caso de empeorar se acepta con una probabilidad $w = e^{-\delta/t}$ donde δ denota la diferencia entre la nueva diferencia simétrica global y la anterior a la excitación. Este modelado permite ir seleccionando los mejores resultados para cada cara escogida aleatoriamente y a la vez, aprobar resultados que no son buenos pero que permitirán la exploración del espacio de

soluciones evitando caer en mínimos locales. El porcentaje de aceptados oscila entre el $13-17\,\%$ como se puede apreciar en la tabla 2.

Se pueden visualizar las ejecuciones de este experimento en: SA Classic livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.095	0.06	0.044591	0.002994	0:00:24.935933	0.167	0.833	10
5	0.095	0.03	0.041228	0.002983	0:00:49.987519	0.146	0.854	10
5	0.95	0.06	0.035595	0.001877	0:07:31.303465	0.137	0.863	10
5	0.15	0.06	0.045873	0.0023	0:00:28.202458	0.163	0.837	10
5	0.15	0.03	0.039684	0.001464	0:00:53.721521	0.144	0.856	10

Cuadro 2: Resultados SA Classic

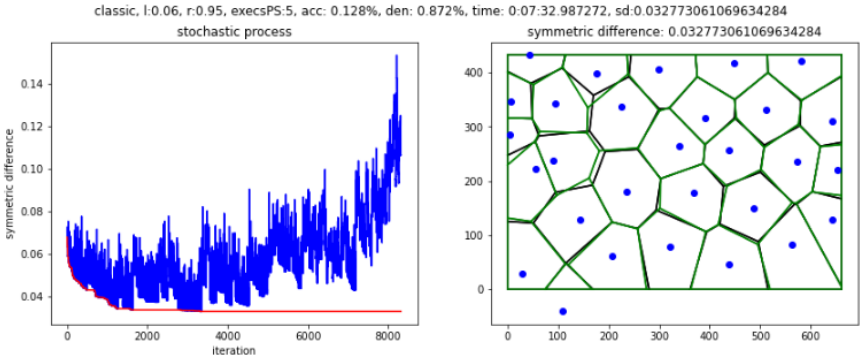

Figura 24: Evolución de SA Classic con r=0.95 y l=0.06

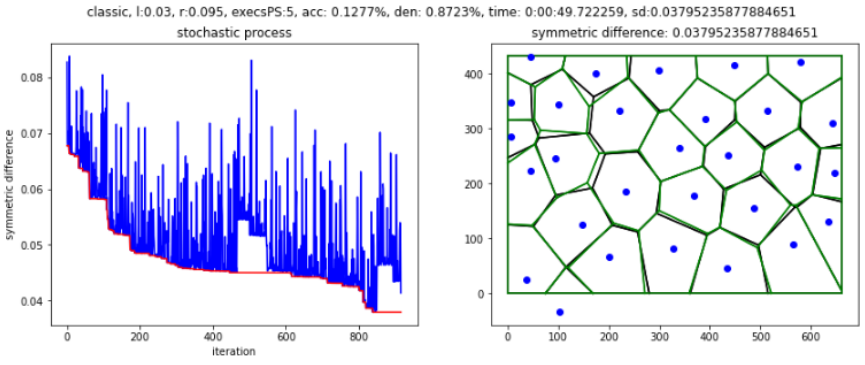

Figura 25: Evolución de SA Classic con r=0.095 y l=0.03

4.2. simulated Annealing Peers

En esta perspectiva se escogerá una cara aleatoriamente y a continuación, al azar, una cara vecina a la escogida. Se excitará el generador de la cara principal y se estudiará la diferencia simétrica de las dos caras. En el caso de la aceptación por conjunción (and) si ambas mejoran la solución es aceptada, en caso de la aceptación por disyunción (or) si una de las dos mejora, en caso contrario, para las dos, se acepta con una probabilidad $w = e^{-\delta/t}$, dónde δ denota la diferencia entre la suma de las nuevas diferencias simétricas locales y la suma de las anteriores a la excitación.

4.2.1. And

La aproximación puntualmente da resultados muy buenos como se puede apreciar en la figura 26, pero no de forma general. Se trata de una política muy exigente ya que tiene que mejorar la energía local de las dos caras. Se espera que el porcentaje de aceptados sea bajo, como se puede apreciar en la tabla 3 oscila entre el $7 - 9\,\%$. Son comunes los estancamientos como consecuencia de la dificultad de la aceptación, véase la figura 27. Las diferentes ejecuciones de este experimento se encuentran en: SA Peers And livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.095	0.03	0.051352	0.003616	0:00:24.293849	0.082	0.918	10
5	0.095	0.06	0.055939	0.003665	0:00:13.323833	0.088	0.912	10
5	0.15	0.03	0.052142	0.003182	0:00:26.381752	0.078	0.922	10
5	0.15	0.06	0.050063	0.005311	0:00:13.423872	0.083	0.917	10
5	0.95	0.06	0.042492	0.002658	0:03:26.338822	0.092	0.908	10

Cuadro 3: Resultados SA Peers and

4.2.2. Or

La diferencia simétrica es superior a la de la política anterior como se puede ver en la tabla 4, esto se debe a la tolerancia de la negociación, en cuánto una cara mejora su energía local la solución es aceptada, de ahí el aumento del porcentaje de aceptados y las subidas

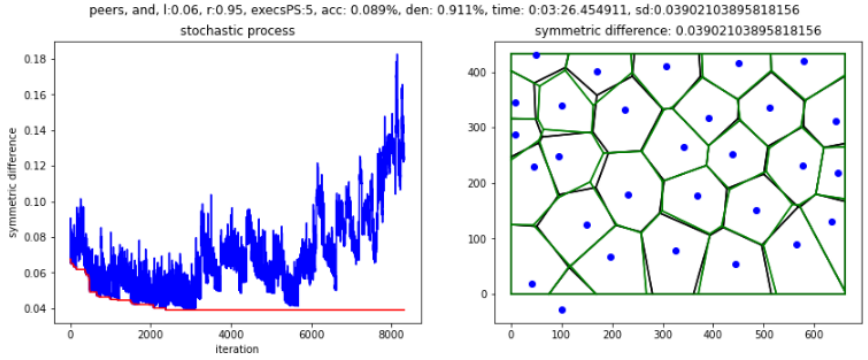

Figura 26: Evolución de SA Peers And con r=0.95 y l=0.06

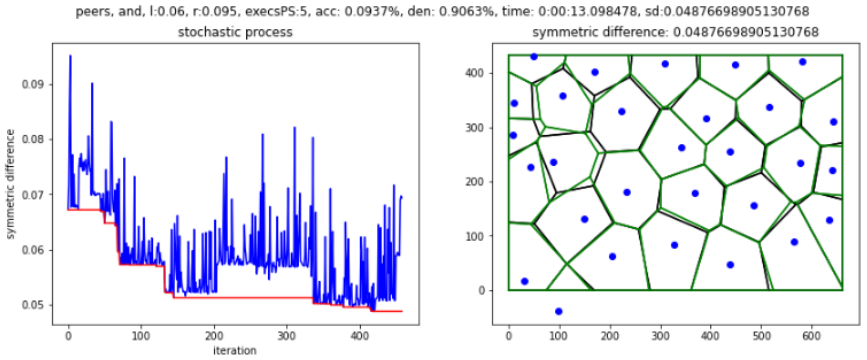

Figura 27: Evolución de SA Peers And con r=0.095 y l=0.06

tan tempranas de la diferencia simétrica en las gráficas de evolución del proceso estocástico (figuras 29 y 30), y el estancamiento en valores muy altos de la diferencia simétrica en iteraciones elevadas (figura 28). El conjunto de ejecuciones quedan plasmadas en SA Peers Or livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.053855	0.002464	0:03:22.985847	0.472	0.528	10
5	0.095	0.03	0.059346	0.003608	0:00:24.063805	0.46	0.54	10
5	0.095	0.06	0.063056	0.003459	0:00:13.153911	0.465	0.535	10
5	0.15	0.03	0.061499	0.00415	0:00:26.035602	0.468	0.532	10
5	0.15	0.06	0.0638	0.00285	0:00:13.215412	0.473	0.527	10

Cuadro 4: Resultados SA Peers Or

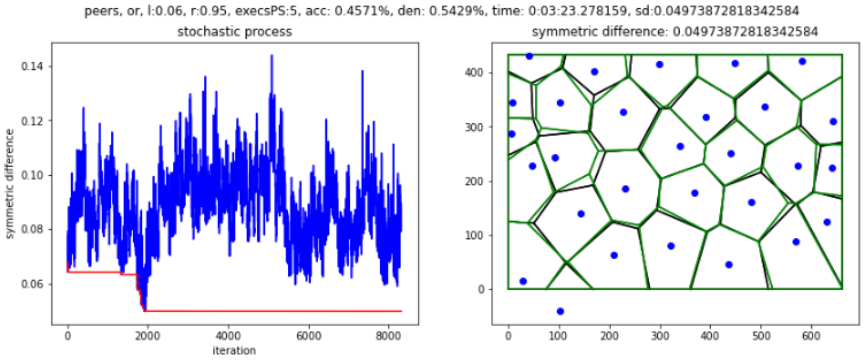

Figura 28: Evolución de SA Peers Or con r=0.95 y l=0.06

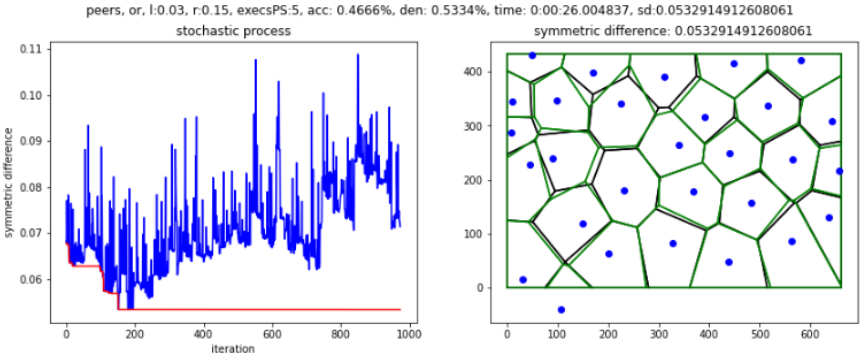

Figura 29: Evolución de SA Peers OR con r=0.15 y l=0.03

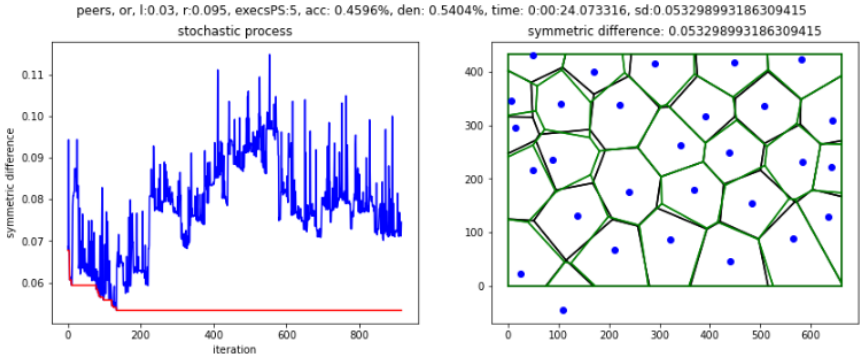

Figura 30: Evolución de SA Peers OR con r=0.95 y l=0.03

4.3. simulated Annealing Groups

En esta perspectiva se escogerá una cara aleatoriamente y todos los vecinos de la misma. Se excitará el generador de la cara principal y se estudiará la diferencia simétrica de todas las caras implicadas. En el caso de la aceptación por conjunción (and) si todas mejoran

la solución es aceptada, en caso de la aceptación por disyunción (or) si una de las caras mejora y en el caso de la aceptación por nº de nodos mejoradas si al menos la mitad de los sujetos mejoran. En caso contrario, para todas las políticas, se acepta con una probabilidad $w = e^{-\delta/t}$ dónde δ denota la diferencia entre la suma de las nuevas diferencias simétricas locales y la suma de las anteriores a la excitación.

4.3.1. And

Se trata de una política muy exigente, más que la política conjuntiva por parejas vista en 4.2.1, ya que el número de implicados siempre es superior. Se espera un porcentaje de aceptaciones muy bajo, como se ve en la tabla 5 oscila entre el $1-3\,\%$. Se aprecian de nuevo los estancamientos vistos en 4.2.1 como consecuencia de la exigencia, véase la figura 32. Para visualizar estas ejecuciones: SA Groups And livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.095	0.06	0.063471	0.002136	0:00:14.187887	0.022	0.978	10
5	0.095	0.03	0.061925	0.00142	0:00:27.929376	0.017	0.983	10
5	0.95	0.06	0.061465	0.002943	0:03:56.422422	0.032	0.968	10
5	0.15	0.06	0.063419	0.002839	0:00:15.477226	0.019	0.981	10
5	0.15	0.03	0.063641	0.002381	0:00:30.169458	0.02	0.98	10

Cuadro 5: Resultados SA groups and

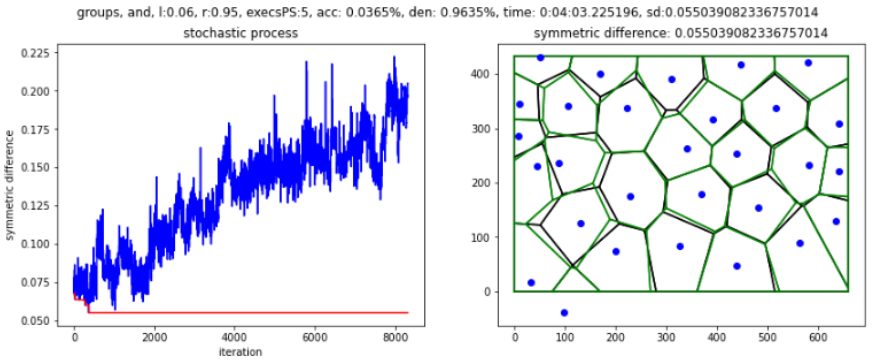

Figura 31: Evolución de SA Groups And con r=0.95 y l=0.06

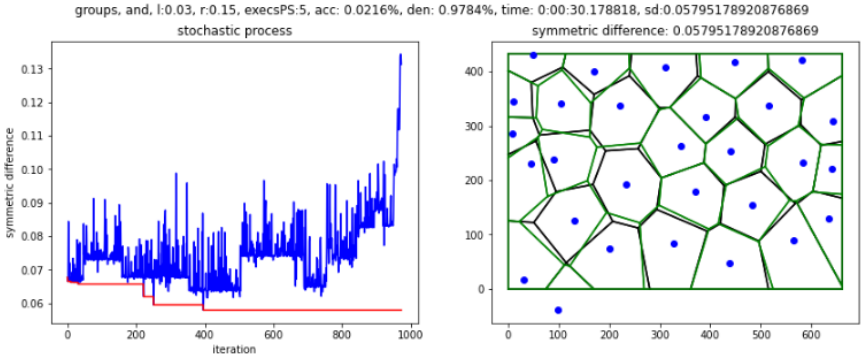

Figura 32: Evolución de SA Groups And con r=0.15 y l=0.03

4.3.2. Or

Los resultados medios son muy malos (tabla 6), al igual que la política disyuntiva por parejas (4.2.2) se están aceptando soluciones que no conducen al *annealing* a disminuir la energía del sistema. Se acepta en una frecuencia muy alta la solución propuesta ya que casi siempre mejora alguno de los agentes implicados, existe un porcentaje de aceptados de entre $88 - 90\,\%$. Como se observa en la figura 34 en las primeras iteraciones se pierde la configuración ventajosa que aportaba la colocación de los generadores en los baricentros de los polígonos resultantes de la salida del algoritmo Planting Seeds. Una vez perdida, en iteraciones superiores, el *annealing* no consigue recuperarse, véase la figura 33. Para ver otras ejecuciones: SA Groups Or livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.067495	0.000432	0:03:41.563245	0.906	0.094	10
5	0.15	0.03	0.066575	0.002216	0:00:29.368258	0.892	0.108	10
5	0.15	0.06	0.067298	0.001073	0:00:14.958717	0.895	0.105	10
5	0.095	0.03	0.067572	0.000285	0:00:27.319656	0.89	0.11	10
5	0.095	0.06	0.067045	0.000794	0:00:13.882258	0.885	0.115	10

Cuadro 6: Resultados SA groups or

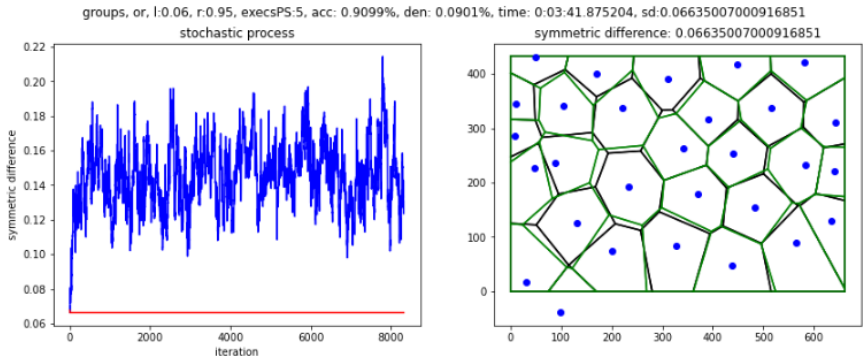

Figura 33: Evolución de SA Groups Or con r=0.95 y l=0.06

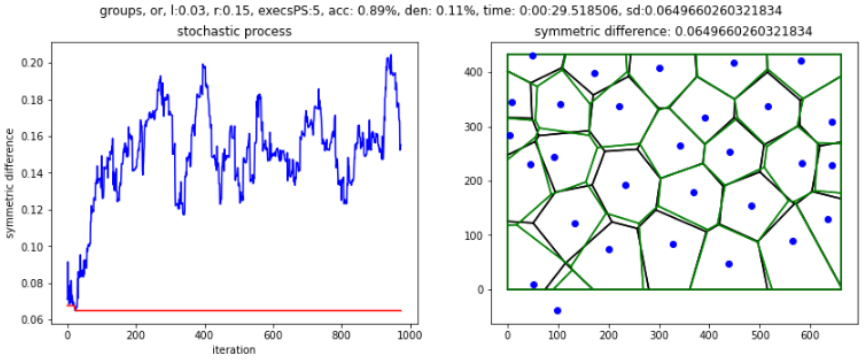

Figura 34: Evolución de SA Groups Or con r=0.15 y l=0.03

4.3.3. Numbers

Estos resultados son muy buenos (tabla 7), se explota correctamente la propiedad de localidad permitiendo pasar solo aquellas soluciones que mejoran la mayoría de las energías de los agentes y dando una segunda oportunidad en función de una probabilidad $w = e^{-\delta/t}$, asegurando el no estancamiento en mínimos locales. La proporción de aceptados oscila entre entre el $19 - 21\,\%$, similar al SA Classic $(13 - 17\,\%)$. Las evoluciones estocásticas que se pueden ver en las figuras 35 y 36 son muy similares a las de las figuras 24 y 25. Todas estas ejecuciones pueden ser encontradas en: SA Groups Numbers livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.039643	0.001304	0:03:46.998677	0.216	0.784	10
5	0.095	0.03	0.04778	0.005827	0:00:28.111056	0.2	0.8	10
5	0.095	0.06	0.049239	0.005431	0:00:14.353650	0.201	0.799	10
5	0.15	0.03	0.045146	0.003314	0:00:30.313018	0.199	0.801	10
5	0.15	0.06	0.049797	0.003474	0:00:15.912763	0.216	0.784	10

Cuadro 7: Resultados SA groups numbers

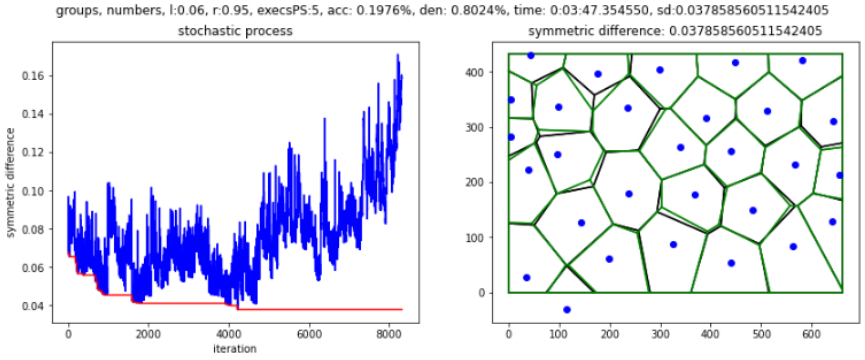

Figura 35: Evolución de SA Groups Numbers con r=0.95 y l=0.06

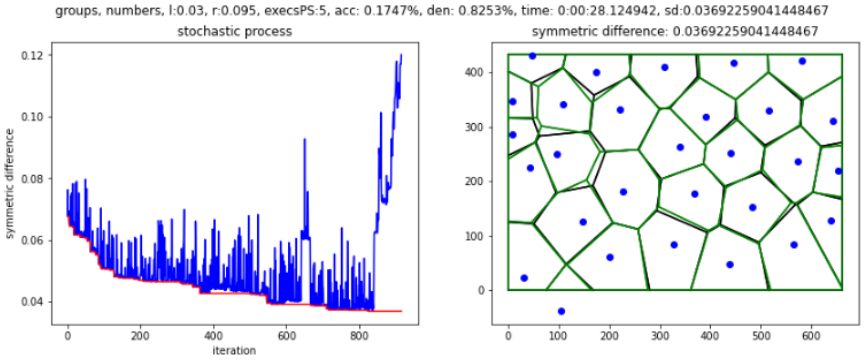

Figura 36: Evolución de SA Groups Numbers con r=0.15 y l=0.03

4.4. simulated Annealing Colours

En esta perspectiva se elegirá una cara principal al azar y todas las caras homólogas de color a la escogida. En el caso de la aceptación por conjunción (and) se aceptará la solución si todas las caras homólogas de color a la elegida mejoran su diferencia simétrica, en el caso de la aceptación por disyunción (or) se aceptará con que solo una mejore y en el caso de la aceptación por nº de nodos mejorados si al menos la mitad mejoran. En caso contrario, para todas las políticas, se acepta con una probabilidad $w = e^{-\delta/t}$ dónde δ denota la diferencia entre la suma de las nuevas diferencias simétricas locales y la suma de las anteriores a la excitación.

4.4.1. And 4Rectangles

Política muy exigente en la que la mitad de la partición debe mejorar, es decir, dos de los cuatro rectángulos en los que ha quedado dividido el espacio por la coloración. Como se aprecia en la tabla 8 sólo se acepta entre el $2-3\,\%$ de las iteraciones. En la figura 38 se aprecian los típicos estancamientos de las políticas por conjunción.Teniendo en cuenta las condiciones, los resultados son aceptables en comparación con el *annealing* clásico visto con anterioridad. Livescript: SA Colours And 4Rectangles livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.095	0.06	0.057741	0.003952	0:00:26.554795	0.028	0.972	10
5	0.095	0.03	0.057823	0.003423	0:00:51.015978	0.022	0.978	10
5	0.15	0.06	0.058722	0.0038	0:00:28.235203	0.024	0.976	10
5	0.15	0.03	0.05692	0.001994	0:00:56.460831	0.021	0.979	10
5	0.95	0.06	0.055136	0.003309	0:07:07.560647	0.03	0.97	10

Cuadro 8: Resultados SA colours and 4rectangles

4.4.2. And Random2Colours

Política muy exigente también en la que todas las caras de la partición coloreadas del mismo modo y distribuidas aleatoriamente deben mejorar. En la figura 40 se aprecian de

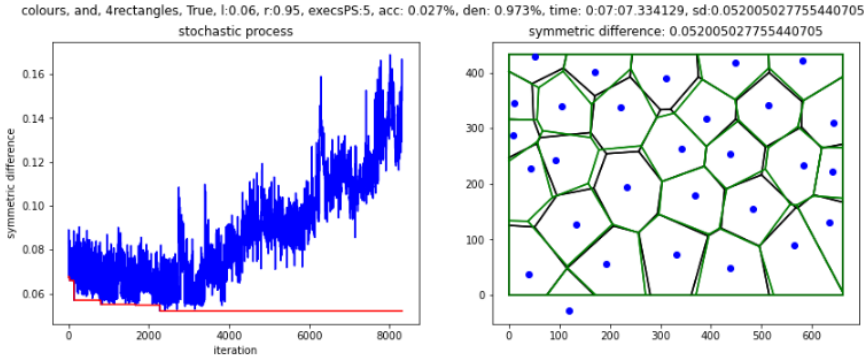

Figura 37: Evolución de SA Colours And 4R con r=0.06 y l=0.95

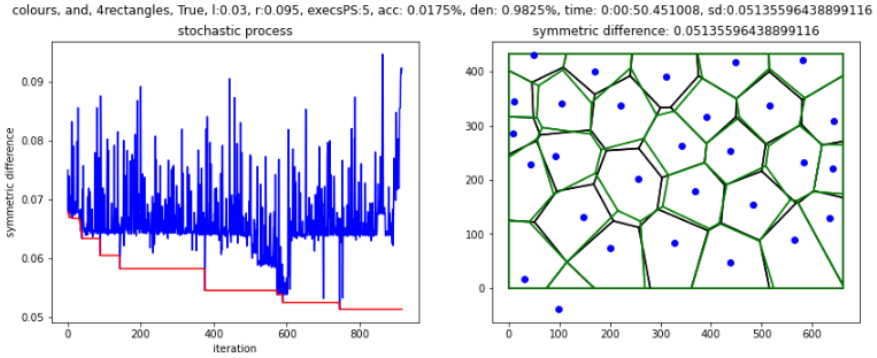

Figura 38: Evolución de SA Colours And 4R con r=0.03 y l=0.095

nuevo los estancamientos de las políticas por conjunción. En este caso la política es un poco menos exigente que la anterior ya que el número de análogos de color puede oscilar, de ese modo se aprecia en la tabla 9 que se acepta entre el $3-4\,\%$ de las iteraciones. Al igual que en la otra coloración los resultados son aceptables en comparación con el *annealing* clásico visto en 4.1.1 teniendo en cuenta las condiciones. Livescript: SA Colours And Random2Colours livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.055941	0.005593	0:07:14.577296	0.036	0.964	10
5	0.15	0.03	0.054394	0.005057	0:00:54.801666	0.035	0.965	10
5	0.15	0.06	0.056046	0.003512	0:00:29.006400	0.037	0.963	10
5	0.095	0.03	0.053529	0.004475	0:00:50.891054	0.04	0.96	10
5	0.095	0.06	0.056961	0.004892	0:00:26.865569	0.04	0.96	10

Cuadro 9: Resultados SA colours and random2colours

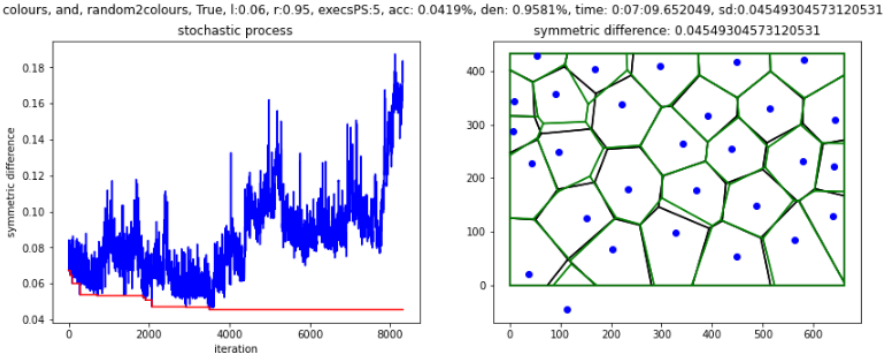

Figura 39: Evolución de SA Colours And R2C con r=0.06 y l=0.95

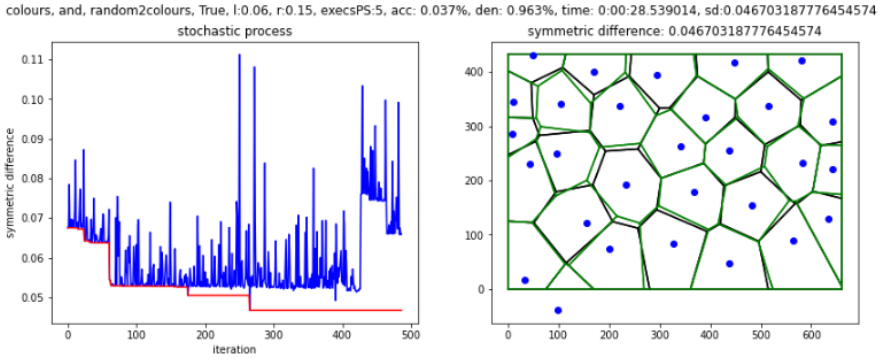

Figura 40: Evolución de SA Colours And R2C con r=0.06 y l=0.15

4.4.3. Or 4Rectangles

Se trata de una política que tolera todo tipo de excitación. Como se puede apreciar en la tabla 10 el porcentaje de aceptación es del 100 % por lo que no conducirá al *annealing* a

reducir la energía del sistema, ni si quiera a disminuir la energía inicial (figura 42). Desde primeras iteraciones (figura 42) la energía del sistema se dispara y se explora a partir de este salto en experimentos con mayor número de iteraciones (figura 41). Livescript: SA Colours Or 4Rectangles livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.15	0.03	0.067396	0.000512	0:00:40.009387	1.0	0.0	10
5	0.15	0.06	0.067703	5.4e-05	0:00:19.934495	1.0	0.0	10
5	0.95	0.06	0.067398	0.001013	0:04:54.966030	1.0	0.0	10
5	0.095	0.03	0.067684	0.000105	0:00:35.709531	1.0	0.0	10
5	0.095	0.06	0.067231	0.00125	0:00:18.601504	1.0	0.0	10

Cuadro 10: Resultados SA colours or 4rectangles

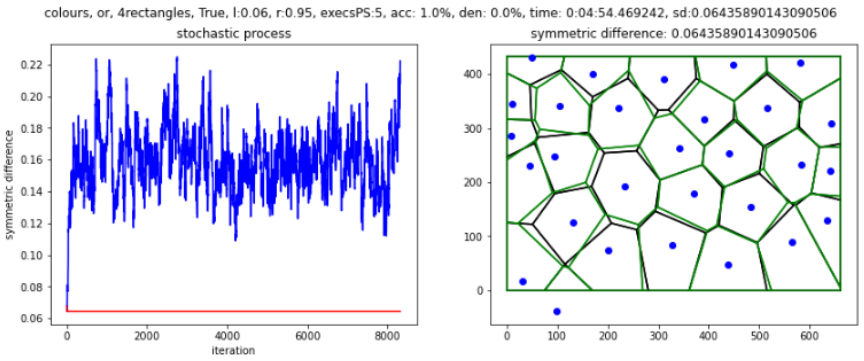

Figura 41: Evolución de SA Colours Or 4R con r=0.06 y l=0.95

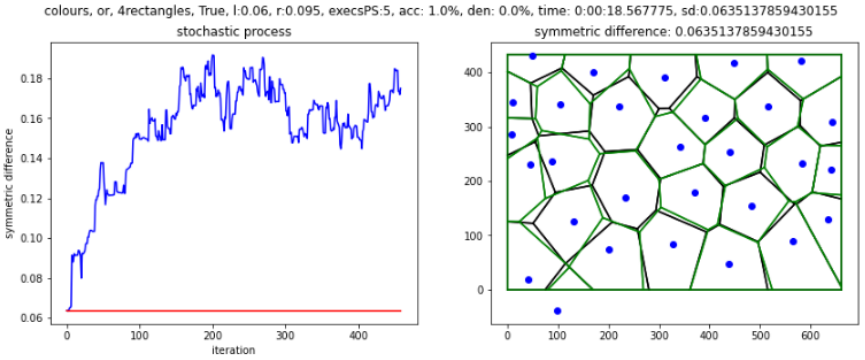

Figura 42: Evolución de SA Colours Or 4R con r=0.06 y l=0.095

4.4.4. Or Random2Colours

Al igual que la política anterior (4.4.3), con un porcentaje de aceptación del 100 % (tabla 11) de nuevo no se conduce correctamente el *annealing* (figuras 43 y 44). Livescript: SA Colours Or Random2Colours livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.15	0.06	0.067407	0.000938	0:00:20.276174	1.0	0.0	10
5	0.15	0.03	0.066991	0.001062	0:00:39.439859	1.0	0.0	10
5	0.095	0.06	0.067501	0.000482	0:00:18.660949	1.0	0.0	10
5	0.095	0.03	0.067704	6.5e-05	0:00:36.489687	1.0	0.0	10
5	0.95	0.06	0.06773	1.3e-05	0:04:57.548754	1.0	0.0	10

Cuadro 11: Resultados SA colours or random2colours

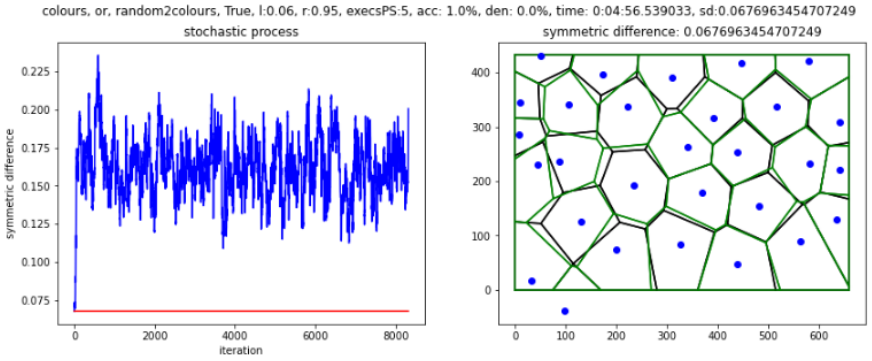

Figura 43: Evolución de SA Colours Or R2C con r=0.06 y l=0.95

4.4.5. Numbers 4Rectangles

Política muy tolerante también en la que se aceptan el 99 % de las iteraciones (tabla 12), de nuevo no se guiará de forma eficiente al *annealing* (figuras 45 y 46). Resultados en torno al 6,7 % de diferencia simétrica, lejos del $3{,}5 - 4{,}5$ % del clásico (4.1.1). Livescript: SA Colours Numbers 4Rectangles livescript.

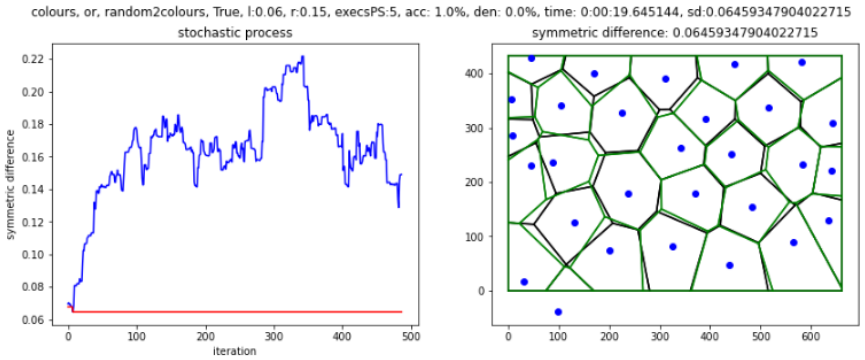

Figura 44: Evolución de SA Colours Or R2C con r=0.06 y l=0.15

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.06756	0.000517	0:07:07.470074	0.994	0.006	10
5	0.095	0.06	0.067411	0.000872	0:00:26.509647	0.992	0.008	10
5	0.095	0.03	0.067459	0.000572	0:00:50.298437	0.993	0.007	10
5	0.15	0.06	0.067462	0.000765	0:00:28.193392	0.992	0.008	10
5	0.15	0.03	0.067491	0.000737	0:00:56.931096	0.994	0.006	10

Cuadro 12: Resultados SA colours numbers 4rectangles

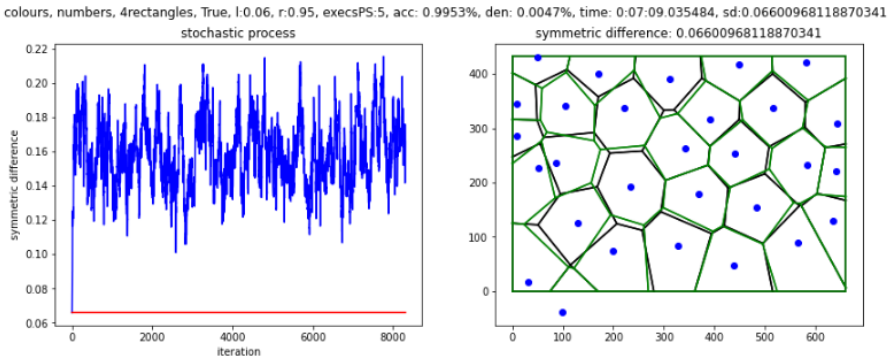

Figura 45: Evolución de SA Colours Numbers 4R con r=0.06 y l=0.95

4.4.6. Numbers Random2Colours

Tolerancia de nuevo del 99 %. Se pueden ver ejemplos de estas evoluciones en las figuras 47 y 48). Al igual que la política por división en 4 rectángulos los resultados están muy lejos

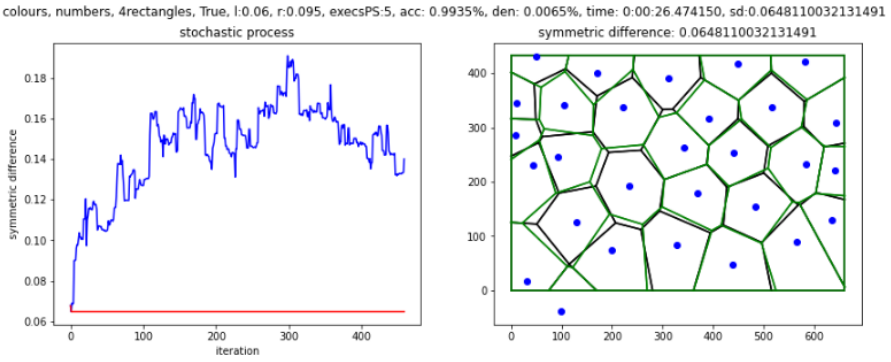

Figura 46: Evolución de SA Colours Numbers 4R con r=0.06 y l=0.095

del *annealing* clásico (tabla 13). Livescript: SA Colours Numbers Random2Colours livescript.

PS	r	l	result	sDev	time	acc	den	n
5	0.95	0.06	0.067382	0.000536	0:07:12.392433	0.997	0.003	10
5	0.095	0.06	0.067554	0.000534	0:00:26.798451	0.996	0.004	10
5	0.095	0.03	0.067385	0.001052	0:00:51.301574	0.997	0.003	10
5	0.15	0.06	0.067371	0.000796	0:00:28.850480	0.997	0.003	10
5	0.15	0.03	0.067165	0.001118	0:00:56.125736	0.998	0.002	10

Cuadro 13: Resultados SA colours numbers random2colours

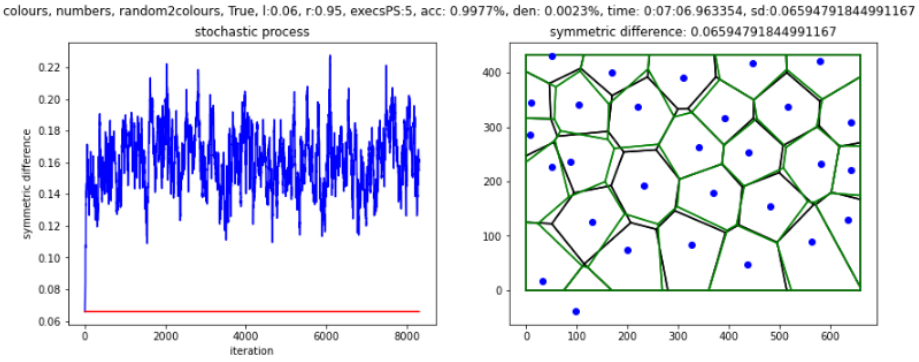

Figura 47: Evolución de SA Colours Numbers R2C con r=0.06 y l=0.95

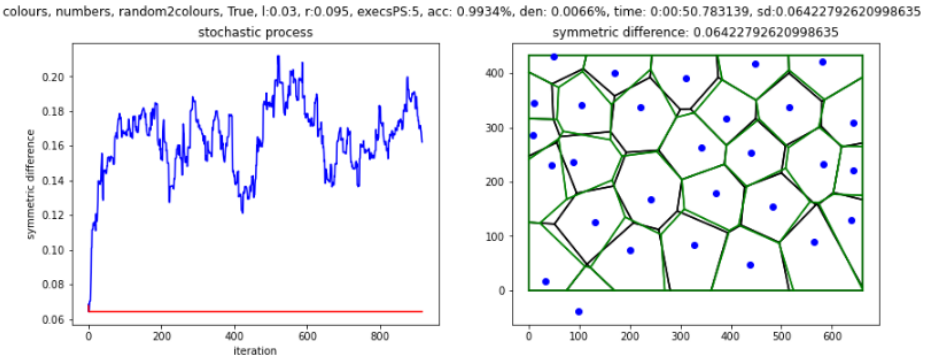

Figura 48: Evolución de SA Colours Numbers R2C con r=0.03 y l=0.095

5. Conclusiones

Todo este trabajo de fin de grado se ha desarrollado a partir del artículo [2], en el que se define una herramienta heurística muy similar a la que aquí se ha construido aunque más compleja. En la aproximación del artículo se emplea un algoritmo de Descenso entre la fase de Planting Seeds y la de simulated Annealing. Este algoritmo parte del conjunto de semillas S generado en el paso anterior e iterativamente se computan excitaciones sobre una de las semillas escogida aleatoriamente de S dentro de un radio r y se acepta la nueva configuración S' si se cumple: $SD_p(S') < SD_p(S)$. Después de un rango de entre 10 a 15 iteraciones los generadores son estables y cualquier otra excitación no supone una mejora, como se puede ver esta aproximación más compleja conforma un punto de salida más ventajoso para el *annealing* que puede conducirle a mejores resultados.

Como se ha visto en las tablas de la sección de resultados (4) se han realizado cinco tipos de ejecuciones para cada política, modificando el parámetro del radio de enfriamiento de la temperatura r para llegar más despacio o más deprisa a la temperatura de congelamiento de las partículas y modificando el parámetro l tratando de explorar con un mayor o menor número de iteraciones cada temperatura y a la vez observar la salida de la función de aceptación ($w = e^{-\delta/t}$) en el caso de la aplicación de la segunda oportunidad.

El resultado que se tiene como referencia pertenece al artículo [2] y se trata de una salida de 0,032 de diferencia simétrica del *annealing* clásico (l = 0,06) con previo *Planting Seeds* y Descenso, para una partición de entrada idéntica a la empleada en este proyecto, véase la figura 49 y compárese con, por ejemplo, la figura 22.

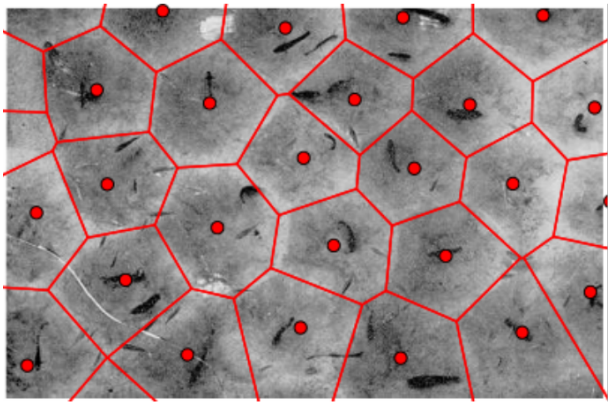

Figura 49: Partición de entrada conformada a partir de la piel del pez Tilapia

Como se puede apreciar en la tabla 14 para la política clásica se obtienen resultados prácticamente idénticos a los del artículo [2], del orden de 0,0327 de diferencia simétrica. Los mínimos globales encontrados para las políticas de grupo por nº de vecinos mejorados y por relación conjuntiva de parejas son muy buenas, de 0,0369225 y 0,039021 respectivamente, mejorando los resultados medios del *annealing* que oscilan entre 0,0356 y 0,0459. Estas políticas se aproximan en gran cantidad a la perspectiva clásica como consecuencia de la explotación de la localidad, en el sentido de que en cada iteración si mejora la energía de la cara escogida, por la propia naturaleza de la teselación de Voronoi también lo hará la de las adyacentes. La diferencia entre la aproximación clásica y las negociaciones anteriores reside en la aceptación que se realiza en el caso de no mejora, por un lado el porcentaje de aceptación por parte de la política conjuntiva por parejas es demasiado restrictiva ($7-9\,\%$) y por otro la política grupal por nº de vecinos se excede en tolerancia ($20-21\,\%$) frente al $13-16\,\%$ que se debería tratar de conseguir (*annealing* clásico). Las políticas conjuntivas por coloración también son aceptables. De este modo, se observa en las tablas de la sección de resultados 14, que las políticas que mejor modelan el guiado son aquellas con una aceptación no superior al $21\,\%$. Todas las políticas mencionadas cumplen con el objetivo del proyecto de tratar de mejorar, aunque esporádicamente, los resultados del *annealing* clásico.

Finalmente, se puede establecer que los resultados del trabajo para la construcción de la herramienta heurística y la investigación del guiado eficiente mediante técnicas de negociación son satisfactorios y que se adecúan a lo esperado. Se deja abierta la línea de investigación para todo aquél que quiera proponer variaciones y/o nuevas políticas contri-

buyendo al desarrollo de la herramienta, siempre tratando de seguir la recomendación de modular la aceptación en torno al $13 - 16\,\%$ (*annealing* clásico).

relationship	method	colourDist	static	PS	r	l	result	acc	den
classic	-	-	True	5	0.95	0.06	0.032773	0.128	0.872
groups	numbers	-	True	5	0.095	0.03	0.0369225	0.175	0.825
peers	and	-	True	5	0.95	0.06	0.039021	0.089	0.910
colours	and	random2colours	True	5	0.15	0.03	0.0438077	0.042	0.958
colours	and	4rectangles	True	5	0.95	0.06	0.0489913	0.030	0.970
peers	or	-	True	5	0.95	0.06	0.049738	0.457	0.543
groups	and	-	True	5	0.95	0.06	0.0550390	0.036	0.964
colours	or	4rectangles	True	5	0.095	0.06	0.063513	1.0	0.0
groups	or	-	True	5	0.15	0.06	0.0641085	0.908	0.092
colours	numbers	random2colours	True	5	0.095	0.03	0.064227	0.993	0.007
colours	or	random2colours	True	5	0.15	0.06	0.064593	1.0	0.0
colours	numbers	4rectangles	True	5	0.095	0.06	0.064811	0.993	0.007

Cuadro 14: Mejores resultados para cada política de annealing por orden ascendente de diferencia simétrica (result)

A. Anexo I: Módulos de la herramienta codificados

La herramienta construida se divide en seis módulos principales:

A.1. DCEL

Este módulo compone e instancia el DCEL que se emplea a lo largo de la herramienta.

A.1.1. Entrada de datos

Como entrada de datos se ha utilizado la partición del artículo [2]. Para ello se han extraído las coordenadas de las aristas y los vértices (relación entre los aristas) a mano, posteriormente se han introducido a la herramienta mediante punteros, se resume en: data.py.

A.1.2. Estructura de datos

Para modelar el DCEL se han creado cuatro clases, arista, vértice, cara y DCEL, con sus propias operaciones. El modelado de las tres primeras se han descrito anteriormente en 2.7.1, la cuarta simplemente es una lista de caras, véase: dcel.py.

A.1.3. Instancia de datos

Para instanciar los datos se llama al constructor de la clase DCEL visto en el apartado anterior, previamente se reescalan todas los vértices de la partición con el objetivo de ajustarlo a deseo del usuario, revise dcelInstance.py.

A.2. simulated Annealing

Este módulo compone el optimizador del ajuste del Voronoi inverso con las diferentes técnicas de negociación y aceptación.

A.2.1. Diferencia simétrica

Componente encargado del cálculo de la diferencia simétrica local de las diversas políticas de negociación presentadas y de la posterior energía global, symmetricDifference.py.

A.2.2. Clásico

Para la construcción de la perspectiva del annealing clásico, resumido en: simulatedAnnealing.py.

A.2.3. Por parejas

Para la construcción de la perspectiva del *annealing* con relación de pareja entre dos vecinos adyacentes.

Por disyunción simulatedAnnealingPeers.py - método or.

Por conjunción simulatedAnnealingPeers.py - método and.

A.2.4. Por comunidades de vecinos

Para la construcción de la perspectiva del *annealing* con relación de comunidad.

Por disyunción simulatedAnnealingGroups.py - método or.

Por conjunción simulatedAnnealingGroups.py - método and.

Por nº de mejoras simulatedAnnealingGroups.py - método numbers.

A.2.5. Por comunidades de coloración

Para la construcción de la perspectiva del *annealing* con relación de coloración.

Por disyunción simulatedAnnealingColours.py - método or.

Por conjunción simulatedAnnealingColours.py - método and.

Por nº de mejoras simulatedAnnealingColours.py - método numbers.

A.3. Planting Seeds

Módulo encargado de las ejecuciones del algoritmo *Planting Seeds*, de la colocación inicial de generadores en los polígonos de la partición y de la coloración de los mismos para conformar las comunidades por coloración.

Coloración plantingSeeds.py - distroPoints.

Primera iteración de Planting Seeds plantingSeeds.py - pS

Siguientes iteraciones de Planting Seeds hasta n plantingSeeds.py - pScont

A.4. Tools module

Módulo herramienta o de *tooling* para realizar los cálculos geométricos vistos en 2.8, toolsModule.py.

A.5. Plotting module

Módulo encargado de la customización de la visualización gráfica que se ha visto a lo largo de la memoria y que se muestra en los *livescripts* entregados, plottingModule.py.

A.6. Data processing

Componente cuya función es el procesado de datos de las ejecuciones, generando ficheros en formato .csv que resumen las ejecuciones que posteriormente se han exportado con Latex, y componiendo los *livescripts* de los ejemplos de las ejecuciones y de la evolución de las mismas. dataProcessing.py.

B. Anexo II: Presupuesto

Se trata de un proyecto que no requiere de unas especificaciones técnicas altas, con un portátil y tecnología de virtualización se ha podido construir la herramienta. Se resumen los costes en la tabla 15. Python 3 es el lenguaje con el que se ha desarrollado, como entorno de programación se ha empleado Anaconda Navigator, que a la vez sirve de gestor de entornos, paquetes y dependencias Python sin la necesidad de emplear la línea de comandos. Entre las bibliotecas utilizadas se destacan: *math*, *numpy*, *shapely*, *scipy*, *matplotlib* y *simpy*. Para la posterior distribución de la herramienta se ha escogido la tecnología de virtualización Docker, que facilita la paquetización de dependencias y código así como el despliegue de la aplicación en cuestión. Para la visualización final se ha empleado Jupyter Notebook que permite la entrega de los diferentes tipos de ejecuciones en formato *live script*. Para la realización del documento se ha empleado el sistema de composición de textos Latex, a través de la plataforma Overleaf. Para la construcción de las imágenes mostradas se ha empleado el software de diseño asistido Autocad y el de edición de imágenes Photoshop, ambas edición prueba.

Tecnología	**Coste**
Dell Latitude 7490	1000 €
Ubuntu 16.04 LTS	0 €
Anaconda Navigator	0 €
Docker	0 €
Jupyter-Notebook	0 €
Overleaf, Latex	0 €
AutoCAD	0 €
Photoshop	0 €
Total	1000 €

Cuadro 15: Tecnología y coste de la misma

C. Anexo III: Manual de usuario

C.1. Gestión del repositorio

La distribución de la herramienta se ha realizado a través de la plataforma Github, el repositorio es cedelasen/react.

C.1.1. Clonación del repositorio

En primer lugar se clona el repositorio en nuestra consola preferida:

```
cedelasen@debian:~\$ git clone git@github.com:cedelasen/react.git
Cloning into 'react'...
remote: Enumerating objects: 840, done.
remote: Counting objects: 100% (840/840), done.
remote: Compressing objects: 100% (637/637), done.
remote: Total 3948 (delta 272), reused 728 (delta 169), pack-reused 3108
Receiving objects: 100% (3948/3948), 121.51 MiB | 12.60 MiB/s, done.
Resolving deltas: 100% (330/330), done.
```

C.1.2. Contenido del repositorio

Una vez clonado, se cambia de directorio al creado y se observa el contenido del mismo:

```
cedelasen@debian:~/\$ cd react
cedelasen@debian:~/react\$ ls
backup  code  Dockerfile  README.md
```

C.2. Gestión Docker

C.2.1. Contenido de la imagen Docker

Como se observa existe un fichero Dockerfile, este reúne toda la especificación de dependencias necesarias para la ejecución de la herramienta. Se contruye la imagen Docker a partir del Dockerfile del repositorio:

```
cedelasen@debian:~/react\$ docker build -t react:1.0.0 -f Dockerfile .
Sending build context to Docker daemon    303MB
Step 1/9 : FROM debian:latest
 ---> f6dcff9b59af
Step 2/9 : RUN apt-get -qq update &&
apt-get -qq -y install curl bzip2 &&
curl -sSL https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
-o /tmp/miniconda.sh &&
bash /tmp/miniconda.sh -bfp /usr/local &&
rm -rf /tmp/miniconda.sh &&
conda install -y python=3 &&
conda update conda &&
apt-get -qq ...
:::
Successfully built 52e76a413d89
Successfully tagged react:1.0.0
```

C.2.2. Construcción del contenedor Docker

Una vez creada la imagen Docker, se levanta en modo interactivo el contenedor en la red del host y montando como volumen el directorio con el código del repositorio y los resultados obtenidos:

```
cedelasen@debian:~\$ docker run --network host -v \$(pwd):/app -it react:1.0.0
(react) root@debian:/app#
```

C.2.3. Ejecución de Jupyter-Notebook

Posteriormente, con la consola del contenedor ya abierta, se ejecuta el comando para abrir el editor y visualizador Jupyter Notebook.

```
(react) root@debian:/app\$ jupyter-notebook --allow-root
...
    To access the notebook, open this file in a browser:
        file:///root/.local/share/jupyter/runtime/nbserver-15-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=6b1472a1ab997f79aa0a43c665cd7dd6b94d490f292e3232
     or http://127.0.0.1:8888/?token=6b1472a1ab997f79aa0a43c665cd7dd6b94d490f292e3232
```

Como se puede ver se muestran una serie de urls con autenticación vía token del servicio Jupyter-Notebook que corre en el puerto 8888 del localhost, se pincha en una de las URLs y se abre la interfaz web del servicio en cuestión.

C.3. Interfaz de usuario

La pantalla principal de la interfaz de usuario es la siguiente, como se puede observar hay dos directorios, uno de backup y otro dónde está guardado el código, y dos ficheros, el README.md del repositorio y el Dockerfile con las dependencias necesarias:

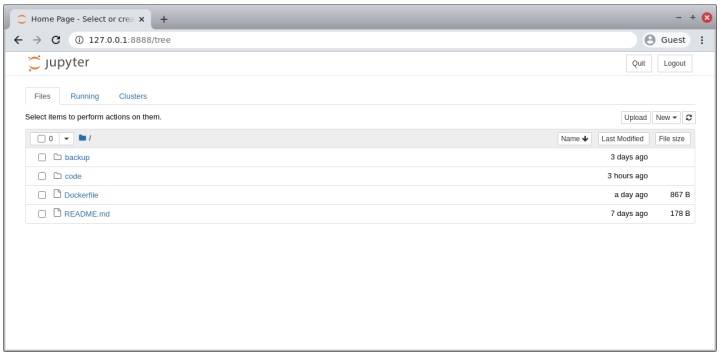

Figura 50: Pantalla principal interfaz de usuario

C.3.1. Dockerfile

Si en la pantalla anterior se clicka en Dockerfile se puede observar el fichero de contenerización de dependencias con las directivas FROM (para establecer la imagen base), RUN (para ejecutar el comando especificado), ENV (para establecer variables de entorno), WORKDIR (para fijar el directorio de trabajo) y ENTRYPOINT (para asignar la orden de arranque del contenedor):

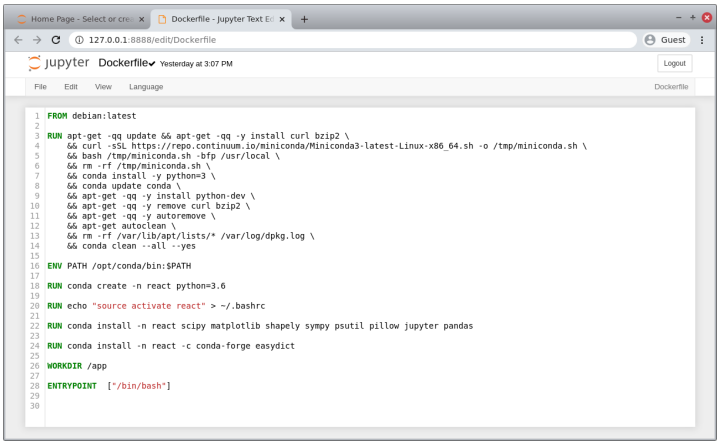

Figura 51: Especificación del Dockerfile del proyecto

C.3.2. Carpeta principal

Si en la pantalla principal se clicka en code se accede al directorio dónde se almacena todos los módulos de la herramienta vistos en A, ficheros de salida de la herramienta (directorio out), ficheros temporales (directorio tmp) y *livescripts* con los resultados y resúmenes de las ejecuciones.

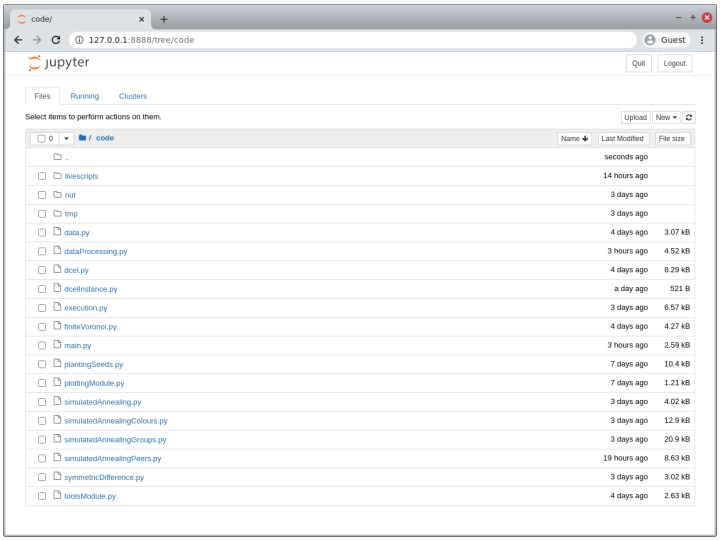

Figura 52: Contenido del directorio Code

C.3.3. Ejemplo de ejecución de la herramienta

Si se quiere ejecutar la herramienta primero se debe editar el fichero *main.py* para introducir los parámetros deseados. Se puede observar un diccionario (estructura de datos de clave-valor) en la función de procesado de argumentos (figura 53) con los datos que hay que modificar. El tipado de datos aceptado es:

Ejecuciones de la herramienta, i Cualquier entero positivo.

Ajuste de las iteraciones por temperatura, l Cualquier float positivo.

Radio de enfriamiento, r Cualquier float positivo.

Ejecuciones Planting Seeds, maxExecs Cualquier entero positivo.

Mínimo en la creación de aleatorios, minR No tocar.

Máximo en la creación de aleatorios, maxR No tocar.

Relación entre nodos, relationship Valores aceptados: “classic”, “peers”, “groups”, “colours”.

Técnica de aceptación, method Valores aceptados: “and”, “or”, “numbers”

Distribución de color, colourDistribution Valores aceptados: “4rectangles”, “random2Colours”

Estaticidad de los colores, static No tocar.

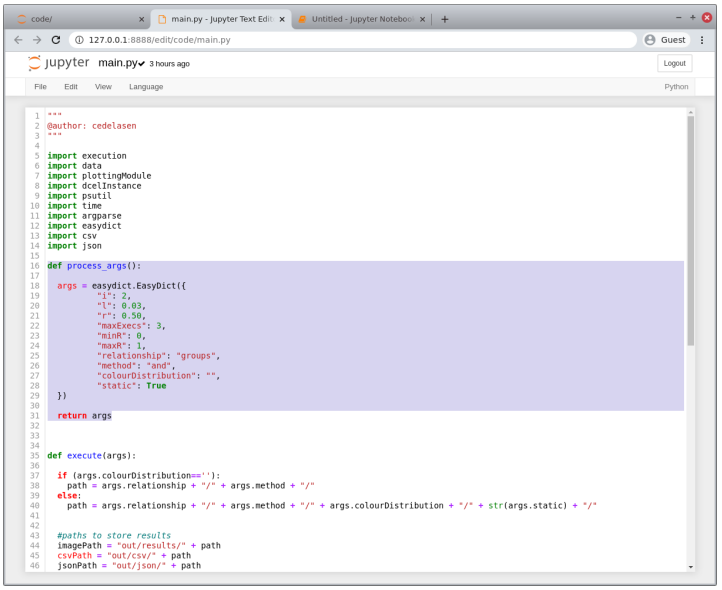

Figura 53: Establecimiento de parámetros en fichero main.py

Posteriormente, en la pantalla del apartado C.3.2 se abre una consola Python3 como se aprecia en la figura 54, se ejecuta el script principal de la herramienta como en la imagen 55 y se observa la salida de la herramienta.

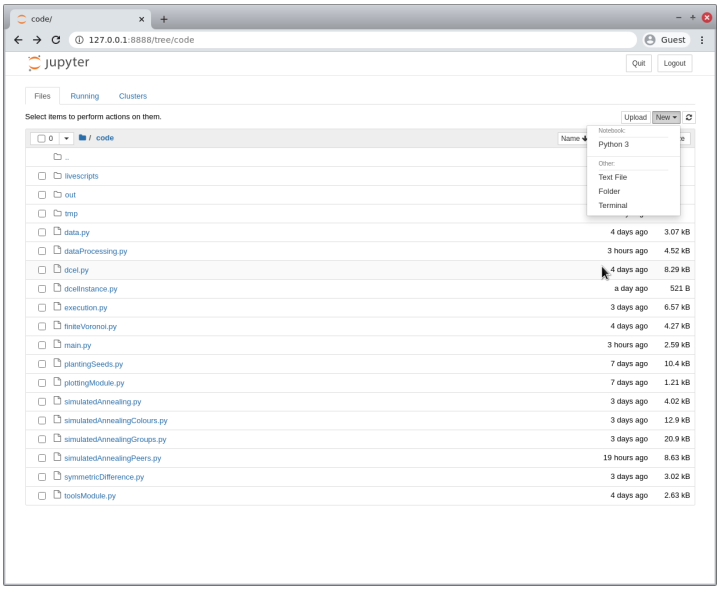

Figura 54: Apertura de consola del tipo Python3

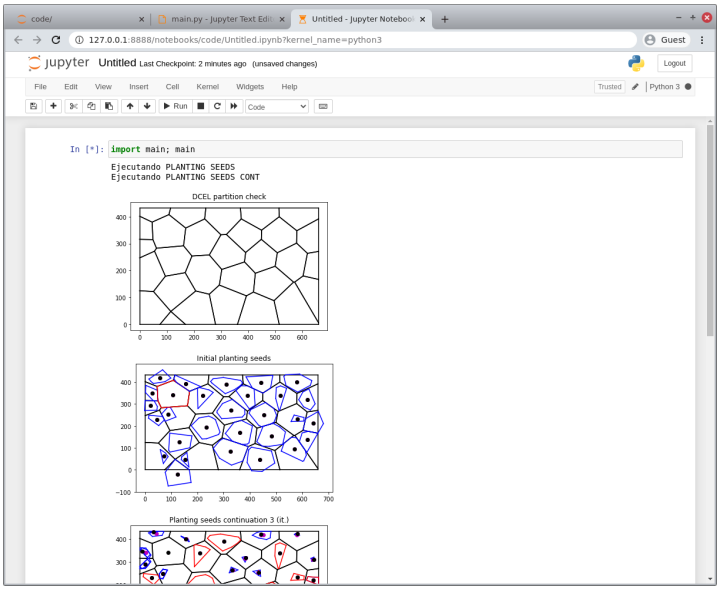

Figura 55: Ejemplo de la salida de una ejecución de un experimento tras establecimiento de parámetros en main.py

Finalmente si se desea obtener un resumen de las ejecuciones para una determinada técnica de negociación se edita el fichero *dataProccessing.py* mostrado en la figura 52 con los argumentos del apartado C.3.3. Un ejemplo de ejecución se muestra en la figura 56.

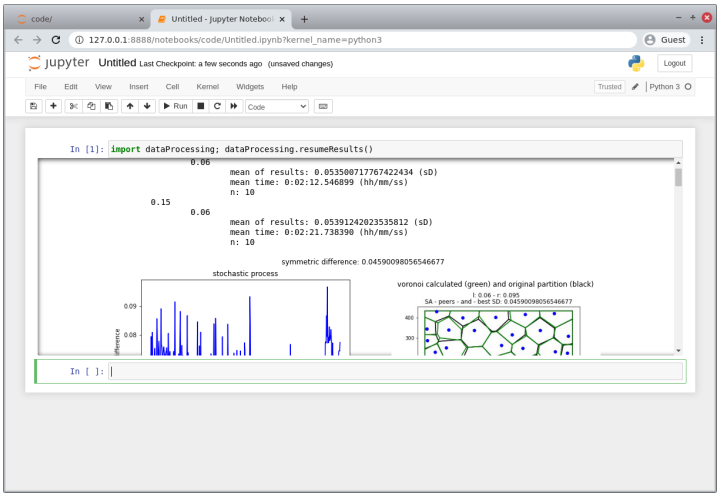

Figura 56: Interfaz de procesamiento de datos

Bibliografía

[1] Wikipedia. (2020) Euler diagram of the different classifications of metaheuristics. [Online]. Available: https://en.wikipedia.org/wiki/Metaheuristic

[2] M. Abellanas and B. Palop, "Fitting voronoi diagrams revisited," *XVII Spanish Meeting on Computational Geometry*, pp. 13–16, 2017. [Online]. Available: https://dmat.ua.es/en/egc17/documentos/book-of-abstracts.pdf

[3] B. Selman and C. Gomes, *Hill-climbing search*, 2006. [Online]. Available: http://www.cs.cornell.edu/selman/papers/pdf/02.encycl-hillclimbing.pdf

[4] M. G. Resend and C. C. Ribeiro, "Greedy randomized adaptive search procedures," pp. 1–4. [Online]. Available: http://www.optimization-online.org/DB_FILE/2001/09/371.pdf

[5] C. Blum, "Blum, c.: Ant colony optimization: Introduction and recent trends. phys. life reviews 2, 353-373," *Physics of Life Reviews*, vol. 2, pp. 353–373, 12 2005. [Online]. Available: https://www.ics.uci.edu/~welling/teaching/271fall09/antcolonyopt.pdf

[6] Y.-D. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, pp. 1–38, 01 2015. [Online]. Available: http://mat.uab.cat/~alseda/MasterOpt/PSO-ZWJ.pdf

[7] F. Neri and C. Cotta, *A Primer on Memetic Algorithms*, 01 2012, vol. 379. [Online]. Available: http://www.lcc.uma.es/~ccottap/papers/MA-primer.pdf

[8] A. Dobrin, "A review of properties and variations of voronoi diagrams," 2005. [Online]. Available: https://www.whitman.edu/Documents/Academics/Mathematics/dobrinat.pdf

[9] O. Erdinc, *Optimization in Renewable Energy Systems*. Butterworth-Heinemann, 2017. [Online]. Available: https://www.sciencedirect.com/book/9780081010419/optimization-in-renewable-energy-systems

[10] C. Grima. (2012, 01) ¿está voronoi? que se ponga. [Online]. Available: http://naukas.com/2012/01/28/esta-voronoi-que-se-ponga/

[11] ——. (2011, 12) Cada uno en su región y Voronoi en la de todos. [Online]. Available: http://naukas.com/2011/12/23/cada-uno-en-su-region-y-voronoi-en-la-de-todos

[12] E. F. Morales, "Capítulo 6 - Recocido Simulado," 2007. [Online]. Available: https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/sa.pdf

[13] F. Blanco-Silva. (2014, 10) Computational Geometry in Python. [Online]. Available: http://blancosilva.github.io/post/2014/10/28/Computational-Geometry-in-Python.html

[14] Wikipedia, "Doubly connected edge list," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Doubly_connected_edge_list

[15] U. of Arizona, "Computational Geometry - Chapter 2 - Basic Techniques." [Online]. Available: https://www2.cs.arizona.edu/classes/cs437/fall07/Lecture2.pdf

[16] A. Yanguas-Gil. (2017) DCEL implementation in Python. [Online]. Available: https://github.com/anglyan/dcel/blob/master/dcel/dcel.py

[17] M. C. R. Abellón, "Tema 5 - Procesos estocásticos." [Online]. Available: http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/apuntes_procesos.pdf

[18] U. de Sevilla, "Tema 3 - Diagramas de Voronoi." [Online]. Available: http://asignatura.us.es/fgcitig/contenidos/gctem3ma.htm

[19] G. A. Núñez, "Fitting plane tessellations through voronoi diagrams," 10 2016. [Online]. Available: http://oa.upm.es/42883/1/TFG_GUILLERMO_ALONSO_NUNEZ.pdf