Aproximación de imágenes con polígonos mediante triangulación de Delaunay

Guido Dinello
Facultad de Ingeniería
Universidad de la república
Montevideo, Uruguay
guido.dinello@fing.edu.uy

Alexis Baladón
Facultad de Ingeniería
Universidad de la república
Montevideo, Uruguay
alexis.baladon@fing.edu.uy

***Abstract*—Mediante la aplicación de un algoritmo evolutivo se busca generar una representación artística que emule una imágen mediante una red conexa de triángulos a partir de una cantidad fija de vértices.**

***Index Terms*—Algoritmos evolutivos, Procesamiento de Imágenes, Tringulación de Delaunay, Paralelismo, Polígonos**

I. Descripción del problema

La representación de curvas mediante polígonos es una tarea que ha sido revolucionaria en el procesamiento de imágenes. Lograr un adecuado compromiso entre la visualización de una imagen y la complejidad computacional del renderizado es de especial interés en el área de la computación gráfica la cual tiene impacto en diversos sectores como la industria de videojuegos y el diseño gráfico. Si bien este estudio no apunta a abarcar tal amplitud de temas sí se propone lograr y evaluar la aplicación de algoritmos evolutivos para obtener buenas aproximaciones a imágenes utilizando la poligonización por triángulos. Aunque actualmente se han planteado problemas similares donde se intenta generar una imagen con triángulos con cierto grado de transparencia y capaces de superponerse, esta solución intenta aproximar la imagen de entrada sin ninguna de las dos, imponiendo que esta sea formada mediante la unión de triángulos de forma similar a la que una imagen puede ser representada por pixeles cuadrados.

II. Estrategia de resolución

Con el fin de cumplir con el objetivo del problema, se buscará lograr una representación de una imagen usando una red de triángulos conexa y convexa que, a su vez, cumpla la condición de Delaunay [1]. Es decir, que la circunferencia circunscrita de cada triángulo de la red no contendrá a ningún vértice de otro triángulo. Dadas las restricciones del problema, es necesario contar con un método para lograr tales representaciones. En este caso, el algoritmo de Delaunay [1] será capaz de lograr este propósito sirviendo como caja negra para formar una red de triángulos dado un número N de vértices en un orden computacional de $O(N^2)$. Un ejemplo de la aplicación de este algoritmo puede verse en la figura 1

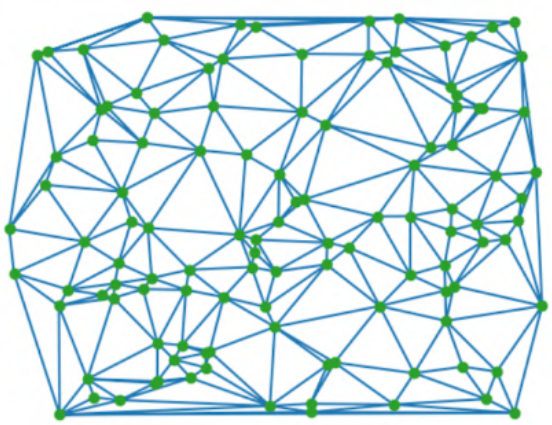


Fig. 1. Resultados de aplicar el algoritmo de Delaunay a 100 puntos aleatorios.

A. Representación de las soluciones

Para este problema la representación elegida es un arreglo de N coordenadas (equivalente a N vértices) que se utilizarán en la imagen poligonal. En particular, cada coordenada (X, Y) será una pareja de enteros que representarán la posición del respectivo punto en la imagen de tamaño (W, H) generada, por lo que el dominio de X se encontrará acotado entre 0 y $W-1$, mientras que Y lo estará entre 0 y $H-1$. El dominio de la representación puede ser expresado como:

$$\mathcal{D} = \{(X, Y) \in \mathbb{Z}^2 \mid X \in [0, W-1], Y \in [0, H-1]\}^N \tag{1}$$

Es importante destacar que estas coordenadas no representan a un triángulo en particular, sino a un vértice, el cuál podrá formar parte de varios triángulos a la vez. Además, el color de cada triángulo no estará especificado en un cromosoma, ya que es tomada la heurística de asignar a un triángulo el color de la imagen original ubicado en el centroide del conjunto de sus vértices, o sea, al triángulo formado por los vértices $\{\vec{v_0}, \vec{v_1}, \vec{v_2}\}$ se le asignará el color de la imagen objetivo en la coordenada $(\vec{v_0} + \vec{v_1} + \vec{v_2})/3$.

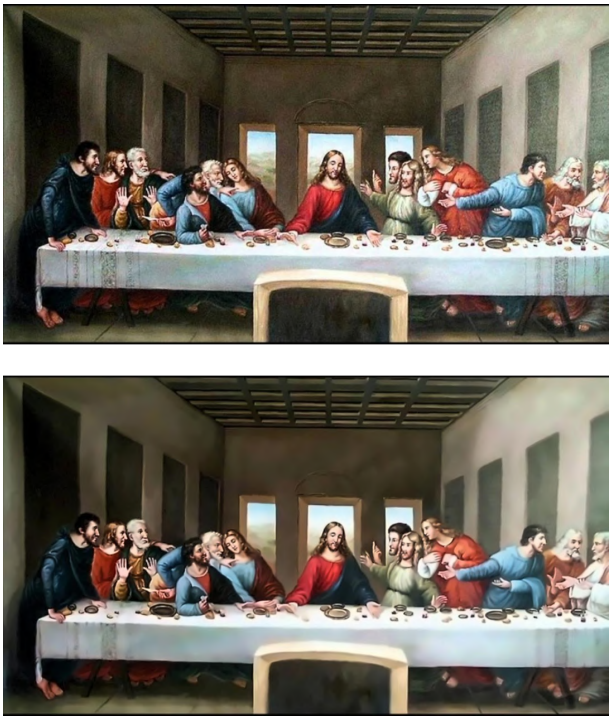


Fig. 2. Resultado de aplicar denoising con NlMeans a imagen utilizada en configuración.

Esta heurística es útil para no incluir a los colores en el genotipo del algoritmo evolutivo, aunque tiene el problema de que sólo se tiene en consideración el color de un único punto que, a causa del ruido, podría contener un color no representativo con respecto a los colores a su alrededor. Aún así, dado lo veloz que esta solución resulta este mecanismo se conserva, aunque para reducir este posible problema se decide utilizar técnicas de denoising con el fin de suavizar la imágen de entrada, haciendo que un individuo tome los colores de la imagen de menor ruido en lugar de la original. Para esto es utilizado el algoritmo fast NlMeans a color [2] proveído por la librería Open-CV [3]. En la figura 2 puede observarse un ejemplo de aplicar este algoritmo a la imagen utilizada en la configuración.

Adicionalmente, se toma la decisión de fijar 4 vértices en las esquinas de la imagen (estando estos fuera de la representación), dado que en caso contrario existirían secciones de la imagen sin un color definido. Estos 4 vértices claramente no formarán parte de la representación, por lo que el algoritmo requerirá un mínimo de 5 vértices para ser ejecutado.

Debe destacarse que este tipo de representación admite que existan varios fenotipos iguales para distintos genotipos, lo cuál no es un punto a favor para esta. Sin embargo, se tomarán medidas en la inicialización para mitigar este problema de las cuales se hablará más adelante.

B. Función de fitness

La función de fitness a utilizar será el error cuadrático medio entre la imagen generada y la imagen original ya preprocesada (en este caso mediante denoising).

$$MSE = \frac{1}{WH} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sum_{k=0}^{2} [O(i,j,k) - G(i,j,k)]^2 \quad (2)$$

, donde W es el ancho de la imagen y H su alto.

Se destaca que cada imagen será procesada con formato RGB, por lo cual, como se ve reflejado en la tercera sumatoria, cada pixel estará formado por tres números en el dominio $[0, 255]$.

C. Condiciones de Parada

Será utilizada como condición de parada que se haya alcanzado una cantidad predefinida y máxima de generaciones. En este caso fue tomado un número de generaciones de 100 como parámetro por defecto, momento en el que el algoritmo suele estancarse dado a que, como se verá más adelante, tiende a no continuar mejorando dado que está basado en operadores elitistas.

D. Operadores Evolutivos

1) Mutación:

a- Gausiana : Este operador suma un determinado valor aleatorio con distribución normal con cierta media y desvío a un grupo aleatorio de alelos. Si bien el operador se basa en esta política, se decide modificarlo levemente con el fin de ajustarse a los requerimientos del problema. En este caso, se utilizarán los valores de media $\mu = 0$ y un desvío σ que, en vez de tomar un valor fijo, tomará el máximo valor del gen mutado multiplicado por una constante a la cual de aquí en adelante se le llamará gaussian rate. Esto se realiza dado que el valor máximo que puede tomar un gen corresponde a $W - 1$ al estar este en un locus de índice par y a $H - 1$ en caso contrario.

Por otro lado, dado que un alelo de un individuo está formado por un valor de una coordenada de los vértices en su fenotipo, tiene más sentido mutar cada valor de una coordenada (un gen de tamaño 2) en vez de solo uno, lo cual ya no sería equivalente a desplazar un vértice únicamente vertical u horizontalmente. Consecuentemente, se decide generar una implementación personalizada de este operador para que desplace ambos valores de sus coordenadas, tal como si sus alelos fueran bidimensionales. Es importante destacar además que este es un operador continuo, mientras que un cromosoma está formado por valores enteros, por lo que el valor de la mutación será aproximado a su valor entero más cercano.

De aquí en más se le llamará MUTPB a la probabilidad de mutar un individuo e INDPB a la probabilidad de mutar un gen.

2) Cruzamiento:

- 2PX : Para este operador se utiliza el cruzamiento a dos puntos. Este segmenta a un par de individuos en dos índices y se genera como resultado a dos individuos formados por la combinación de una mitad de cada padre. Este podría llegar a ser altamente disruptivo para la representación planteada, dado que dos individuos con mismos vértices en un distinto orden no serían favorecidos al cruzarse, es más, empeoraría su aptitud, por lo que se tomarán medidas en la inicialización para mitigar dicho problema.

3) Selección:

- Elitista : (Determinista): Se seleccionan a los k individuos de mayor fitness, lo cual asegura la convergencia aunque puede favorecer la deriva genética dada la posible falta de exploración.
- Torneo (Mixto) Dado un grupo de individuos, se creará una cantidad de k torneos para los cuales serán elegidos m individuos al azar, siendo aquel con mejor aptitud seleccionado para el cruzamiento.

Ambas selecciones serán opciones posibles como parámetros de la implementación. La selección por defecto es la del mejor individuo (elitista).

4) Reemplazo:

- $\mu + \lambda$: La solución presentada utiliza una estrategia de reemplazo $\mu + \lambda$, en la cual tanto padres como hijos compiten para ser seleccionados.

E. Inicialización

Una inicialización aleatoria con valores de coordenadas uniformemente distribuidos parece en principio una buena solución. Sin embargo, esto no hace uso de toda la información que una imagen inherentemente provee, por ejemplo, transiciones entre un color y otro (sus bordes). Por esto, se toma la heurística de inicializar un 50% de los vértices del algoritmo de forma aleatoria y otro 50% en bordes de la imagen. Para esto se utiliza el algoritmo Canny [4] proveído por la librería opencv , que dada una imagen retorna un conjunto de vértices. Es importante notar que un 100% sería perjudicial dado que aunque en los bordes se encuentran los cambios más drásticos de colores, imágenes suaves con cambios leves en colores contiguos requerirán eventualmente de un cambio de color de un pixel a otro. Un ejemplo del uso de este algoritmo es el de la figura 3

Fig. 3. Resultado de aplicar detección de bordes con Canny a imagen utilizada en configuración.

Luego, como fue comentado, existen varios problemas relacionados a la representación elegida. Entre ellos, problemas de cruzamiento, ya que se espera que este operador no realice cambios excesivos que no favorezcan en lo absoluto al individuo. Por esto, se decide ordenar todos los genotipos de los individuos según el valor de X de cada coordenada y en caso de empate por la coordenada Y. Este ordenamiento sólo se realizará al iniciar el algoritmo, ya que los algoritmos de ordenamiento tienen órdenes computacionales altos, y que de todos modos luego de cruzarse un determinado número de veces los individuos tendrán vértices similares alineados a modo de cruzarse con 2PX sin mayores problemas.

F. Métodos avanzados

El algoritmo planteado puede ser realizado de forma paralela. La arquitectura seguida en este caso es la maestro-esclavo, la cual se basa en distribuir la evaluación del fitness entre varios trabajadores. El algoritmo por lo tanto presenta un parámetro de cpu_count para regular la cantidad de procesadores siendo utilizados en simultáneo.

G. Parámetros

A continuación se encuentran los parámetros involucrados en el algoritmo, junto con sus valores por defecto o no para aquellos que es necesario que el usuario introduzca para ejecutar la solución.

III. Evaluación Experimental

Debido a la naturaleza estocástica de los algoritmos que fueron utilizados es necesario realizar una evaluación estadística que nos permita realizar inferencias sobre su desempeño. Es entonces el objetivo de esta sección, el exponer la metodología empleada así como los resultados obtenidos.

A. Plataforma de Ejecución

Todos los algoritmos referenciados fueron ejecutados con las configuraciones y especificaciones detalladas en la siguiente tabla.

Parametro	Valor por defecto
INDPB	0.1
CXPB	0.9
MUTPB	0.1
NGEN	100
μ	50
λ	50
selection	best
tournament_size	2
gaussian_rate	0.05
input_path	-
input_name	-
width	W
height	H
vertex_count	-
cpu_count	n° de procesadores
edge_rate	0.5

TABLE I
PARÁMETROS Y VALORES POR DEFECTO

Sistema operativo	Windows 10
Procesador	Intel(R) Core(TM) i5-7400
Memoria RAM	8GB
Versión de Python	3.9.6

TABLE II
ESPECIFICACIONES DE HARDWARE

B. Configuración Paramétrica

Bajo la finalidad de encontrar la combinación de parámetros que permitiera al algoritmo evolutivo desempeñarse de mejor manera en el problema abordado, se recopilaron métricas del desempeño del mismo sobre un conjunto independiente de ejecuciones. En particular, para la instancia 15, se reportarán y analizarán los valores de mejor fitness histórico, es decir, el mínimo fitness de una ejecución completa de 100 generaciones sobre 50 individuos para 30 semillas distintas sobre cada una de las combinaciones de parámetros estudiadas. Aún más, sobre los valores mencionados anteriormente se reportarán el mínimo, el promedio y el desvío estándar, así como el tiempo promedio de ejecución para cada configuración sobre las 30 semillas utilizadas.

Las configuraciones estudiadas surgen de la combinación ortogonal de los siguientes conjuntos de parámetros y valores asociados:

1) Probabilidad de Cruzamiento:

$$CXPB \in \{0.8, 0.9\}$$

2) Probabilidad de Mutación:

$$MUTPB \in \{0.01, 0.05, 0.1\}$$

La tabla XVII muestra los valores recabados.

Como primer paso del análisis estadístico realizado se evaluó mediante tests de hipótesis —en particular la prueba bondad de ajuste de Kolmogorov-Smirnov (de dos colas) de la libreria scipy [5] que nos permite comparar la distribución subyacente de una muestra contra otra dada—, si la distribución de los mejores fitness obtenidos en cada ejecución se ajusta a una distribución normal. Los resultados obtenidos se pueden apreciar en las tablas III y IV.

P_c	P_m	min(MSE)	$\overline{MSE}$	$\sigma(MSE)$	$\overline{time}$
0.8	0.01	1247.616	1280.580	17.193	420.956
0.8	0.05	1197.162	1264.674	20.580	425.269
0.8	0.1	1212.100	1261.774	15.694	431.654
0.9	0.01	1241.939	1271.891	18.606	454.564
0.9	0.05	1203.522	1249.126	24.999	456.455
0.9	0.1	1197.937	1242.558	22.956	458.275

TABLE III
MÉTRICAS RECOLECTADAS PARA CADA CONFIGURACIÓN

P_c	P_m	pvalue
0.8	0.01	0.9264
0.8	0.05	0.4760
0.8	0.1	0.9222
0.9	0.01	0.8900
0.9	0.05	0.8486
0.9	0.1	0.7020

TABLE IV
P-VALORES PARA CADA CONFIGURACIÓN

Gráficos de los histogramas asociados se pueden encontrar en el anexo 10.

Como se puede apreciar en la tabla IV, todos los p-valores son mayores a 0.05, asumiendo un nivel de confianza del 95% no podemos rechazar la hipótesis nula por lo que consideramos que nuestros datos se ajustan a una distribución normal.

En segunda instancia, y sirviendonos del resultado de la prueba anterior, se procede a realizar un análisis de varianza (ANOVA [6]) de una vía [3] para determinar la existencia de una diferencia estadísticamente significativa entre las medias de los datos de cada configuración.

$$\begin{aligned} H_0 &: \mu_0 = \mu_1 = ... = \mu_n \\ H_a &: i, j \in \{0, ..., n\} \mu_i \neq \mu_j \end{aligned} \tag{3}$$

Obteniendo el siguiente resultado:

$$p - value = 1.5756 \times 10^{-11}$$

Este resultado —asumiendo nuevamente un nivel de significancia del 0.05—, nos permite sostener la afirmación de que hay al menos dos configuraciones que difieren en su media.

Finalmente, es de interés saber cuáles de estos grupos son los que difieren en su media por lo que se procede a realizar tests post hoc de comparación de la media, más específicamente se llevó a cabo un test de student de cada pareja [7] corrigiendo los p-valores usando el método Holm [8] para controlar el FWER [1].

Los resultados obtenidos se pueden observar en la tabla V.

Como puede observarse en la tabla III, tanto el mejor fitness histórico como promedio es mínimo para la combinación de $[0.9, 0.1]$. Además, por los pvalores obtenidos en la tabla V podemos afirmar la existencia evidencia estadística de que esta combinación es mejor que las demás con un nivel de confianza del 0.05 —a excepción de $[0.9, 0.05]$—. Aún así, se decide tomar $[0.9, 0.10]$ dado que la diferencia en tiempo no es significativa y tuvo mejores resultados en los demás atributos.

Consideramos entonces que la mejor configuración de parametros (dentro de los evaluados) para nuestro algoritmo evolutivo es: $[CXPB, MUTPB] = [0.9, 0.1]$

La imagen utilizada con al configuración obtenida y 10.000 vértices se presenta en la figura 15.

[1] Family-Wise Error Rate [9]

Config1	Config2	pvalue
[0.8 — 0.01]	[0.8 — 0.05]	0.0182
[0.8 — 0.01]	[0.8 — 0.1]	0.0007
[0.8 — 0.01]	[0.9 — 0.01]	0.2794
[0.8 — 0.01]	[0.9 — 0.05]	9.1939×10^{-6}
[0.8 — 0.01]	[0.9 — 0.1]	2.5666×10^{-8}
[0.8 — 0.05]	[0.8 — 0.1]	0.6034
[0.8 — 0.05]	[0.9 — 0.01]	0.4998
[0.8 — 0.05]	[0.9 — 0.05]	0.0857
[0.8 — 0.05]	[0.9 — 0.1]	0.0028
[0.8 — 0.1]	[0.9 — 0.01]	0.1476
[0.8 — 0.1]	[0.9 — 0.05]	0.1476
[0.8 — 0.1]	[0.9 — 0.1]	0.0040
[0.9 — 0.01]	[0.9 — 0.05]	0.0025
[0.9 — 0.01]	[0.9 — 0.1]	2.0608×10^{-5}
[0.9 — 0.05]	[0.9 — 0.1]	0.6034

TABLE V
P-VALORES RESULTADO DEL TEST POST HOC

$tournsize$	min(MSE)	$\overline{MSE}$	$\sigma(MSE)$	$\overline{time}$
2	1277.845	1287.592	18.530	478.451
3	1337.026	1295.067	21.619	491.637

TABLE VI
MÉTRICAS RECOLECTADAS PARA CADA VALOR DE $tournsize$

Fig. 4. ultima_cena.jpg 10000 vértices con configuración obtenida.

C. Análisis Informal

Siguiendo una metodología similar a la mencionada en el inciso anterior realizamos un conjunto de evaluaciones independientes —en concreto se ejecutó el algoritmo sobre 30 semillas distintas—, sobre un parámetro más de nuestro algoritmo.

Utilizando la configuración paramétrica considerada como "óptima" que fue hallada en el apartado que precede, recolectamos valores del desempeño para el algoritmo variando el tipo de selección. Es decir, se pasó de una selección elitista a una por torneo, donde se experimentó con el tamaño del "pool", esto es, la cantidad de individuos seleccionados aleatoriamente de la cual posteriormente se seleccionará al mejor.

Los valores experimentados fueron 2 y 3 y se obtuvieron los resultados correspondientes a las tablas VI, VII. Además se puede encontrar la tabla XVIII del anexo con los resultados completos y gráficos asociados 11.

Analizando los p-valores es razonable decir que ambas muestras provienen de distribuciones normales. Es por esto, que se procede a realizar un test ANOVA que nos proporciona el siguiente resultado :

$$p - value = 1.1232 \times 10^{-15}$$

$tournsize$	pvalue
2	0.8726
3	0.9863

TABLE VII
P-VALORES PARA CADA VALOR DE $tournsize$

Config1	Config2	pvalue
[2]	[3]	0.1628
[2]	elitista	5.2347×10^{-11}
[3]	elitista	4.5051×10^{-12}

TABLE VIII
P-VALORES RESULTADO DEL TEST POST HOC

El resultado obtenido sobre los conjuntos $torneo[2]$, $torneo[3]$ y $elitista$ nos permite concluir que hay una diferencia estadísticamente significativa entre las medias de almenos dos grupos. La continuación del análisis dió como resultado la tabla de p-valores VIII —resultado de realizar un estudio comparativo de a pares como el realizado en la sección Configuración Paramétrica—. Si bien esta no nos permite afirmar que un tamaño del *"pool"* del torneo es mejor que otro si nos permite extraer conclusiones y decantarnos por la selección elitista puesto que muestra resultados superiores tanto en fitness como en tiempo de ejecución.

D. Comparación con otras Técnicas

Para llevar a cabo una correcta evaluación del modelo obtenido es de interés comparar el desempeño del AE frente a otras técnicas de resolución del problema tanto en términos de la calidad de las soluciones alcanzadas como de la eficiencia computacional. Es por esto que se proponen dos variaciones de un algoritmo alternativo.

Algorithm 1 Proposed Alternative

Require: $max\_iter \geq 0$
Require: $max\_eval \geq 0$
Require: $threshold \geq 0$
  $eval \leftarrow evaluate(ind)$
  $i \leftarrow 0$
  $eval\_count \leftarrow 1$
  while $i < max\_iter \wedge eval\_count < max\_evals$ **do**
    $gen = random\_gene(ind)$
    $best\_delta \leftarrow 0$
    $deltas \leftarrow get\_deltas(method, threshold)$
    for $delta$ `in` $deltas$ **do**
      $ind_{gen} \leftarrow ind_{gen} + delta$
      $eval\_count \leftarrow eval\_count + 1$
      $new\_eval \leftarrow evaluate(ind)$
      if new_eval is better than $eval$ **then**
        $best\_delta \leftarrow delta$
        $eval \leftarrow new\_eval$
      end if
      $ind_{gen} \leftarrow ind_{gen} - delta$
    end for
    $ind_{gen} \leftarrow ind_{gen} + best\_delta$
    $i \leftarrow i + 1$
  end while=0

Las dos variantes, denominadas "gaussian" y "local_search" difieren únicamente en la función get_deltas. En el primer caso

devuelve una muestra de valores aleatorios provenientes de una distribución normal de media 0 y desvío igual a $threshold$ y en el segundo todos los valores enteros del rango $[-threshold, threshold]$. Se supone que el primero puede ser interpretado por un un algoritmo genético con una población de un individuo, y probabilidad 1 de mutación. Por otro lado, el segundo sería interpretado como un algoritmo local-search que cambia valores de coordenadas aleatorias en una vecindad de tamaño $threshold$, quedándose siempre con el mejor resultado.

Tanto del algoritmo evolutivo (configurado) como de los dos algoritmos propuestos se llevó a cabo una recopilación de métricas del desempeño de 30 ejecuciones independientes, sobre 3 instancias distintas —"fox"[17], "monalisa"[18] y "old man"[16]—. Los resultados numéricos obtenidos resumidos pueden encontrarse en las tablas IX y X.

Fig. 5. fox.jpg 1500 vértices.

Fig. 6. monalisa.jpg 2000 vértices.

instance	algorithm	min(MSE)	$\overline{MSE}$	$\sigma(MSE)$	$\overline{time}$
fox	EA	194.745	205.305	5.397	144.797
fox	localsearch	167.123	185.915	11.121	348.691
fox	gaussian	189.391	224.928	17.759	359.525
monalisa	EA	128.088	133.316	3.370	212.298
monalisa	localsearch	148.044	165.664	8.011	460.628
monalisa	gaussian	112.576	130.533	14.545	470.093
oldman	EA	593.902	607.164	8.868	326.688
oldman	localsearch	555.127	581.379	12.603	720.360
oldman	gaussian	561.642	618.913	29.671	710.275

TABLE IX
MÉTRICAS RECOLECTADAS PARA CADA PAR ALGORITMO E INSTANCIA

instance	algorithm	pvalue
fox	EA	0.9131
fox	localsearch	0.4604
fox	gaussian	0.8800
monalisa	EA	0.3265
monalisa	localsearch	0.5200
monalisa	gaussian	0.5635
oldman	EA	0.8295
oldman	localsearch	0.4607
oldman	gaussian	0.9711

TABLE X
P-VALORES RESULTADO DEL TEST DE NORMALIDAD PARA UN ALGORITMO E INSTANCIA DADOS SOBRE 30 EJECUCIONES INDEPENDIENTES

Fig. 7. old_man.jpeg 3000 vértices.

Los resultados extensos aparecen en la tabla XIX del anexo y los gráficos 12, 13, 14 sirven de acompañamiento visual.

Observando la tabla X podemos concluir que cada una de las muestras de 30 valores de cada pareja instancia-algoritmo cumplen con la hipótesis de normalidad puesto que los p-valores son todos mayores al nivel de significancia admitido.

Esto nos permite entonces realizar un test de ANOVA para cada instancia, es decir, se evaluará si hay evidencia estadística suficiente para afirmar la existencia de diferencia entre las medias de cada algoritmo en cada una de las instancias en concreto. Los resultados obtenidos se detallan a continuación.

Sobre la instancia "fox" :

$$p - value = 5.8087 \times 10^{-19}$$

Sobre la instancia "monalisa_sqr" :

$$pvalue = 3.2058 \times 10^{-25}$$

Sobre la instancia "old_man" :

$$pvalue = 2.6991 \times 10^{-10}$$

Los 3 resultados anteriores nos indican que efectivamente existen al menos 2 algoritmos que difieren en la media del mejor fitness alcanzado para cada una de las instancias.

Corresponde entonces realizar las pruebas post hoc para identificar entre cuales existe dicha diferencia.

Los resultados se puede apreciar en las tablas XI, XII, XIII.

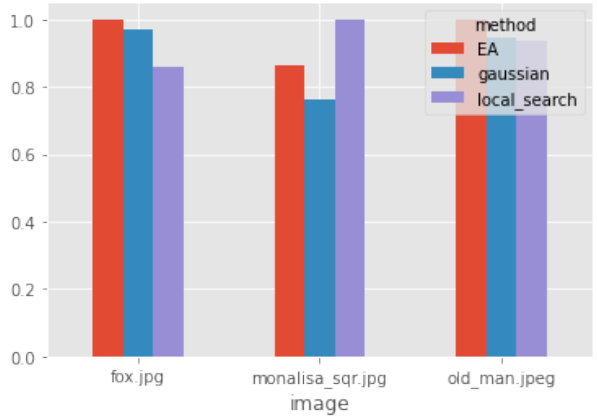


Fig. 8. Mejor fitness (normalizado) para cada instancia.

Respecto a la instancia fox, dado que los p-valores nos permiten comparar las 3 combinaciones concluimos que para dicha instancia el algoritmo con mejor desempeño es la variante localsearch, luego la variante gaussiana y por último el EA. Quizás es prudente remarcar que a pesar del pobre desempeño del EA este logra una diferencia considerable en el tiempo empleado para alcanzar dicho valor, de hecho, este alcanza una solución con aptitud similar en menos de la mitad de tiempo.

En la instancia monalisa podemos contrastar el desempeño de la variante localsearch con el EA y con la gaussiana, pero no a estos últimos dos entre sí. Tanto el EA como el guassiano son más performantes que el localsearch. Una vez más el EA únicamente destaca en el tiempo promedio de ejecución.

Finalmente, para la última instancia podemos extraer conclusiones de las tres comparaciones, donde una vez más el EA se ve superado en la calidad de la soluciones obtenidas por ambas variantes y exclusivamente destaca en el tiempo de ejecución.

Este resultado es desalentador aunque cabe resaltar que el algoritmo evolutivo logra su objetivo en mucho menos tiempo promedio aún dada la cantidad de operaciones realizadas en un algoritmo evolutivo, probablemente debido al uso de múltiples procesadores. Es importante resaltar que debido al elitismo del AE el mejor fitness obtenido generalmente se lograba mucho antes del máximo de generaciones.

Es posible además que exista una configuración de los valores de mutación (Por ejemplo, de un gen o de un individuo) del algoritmo evolutivo para los cuales este tenga un comportamiento similar al algoritmo gaussiano que en varias ocasiones lo supera, por lo cual se puede concluir que aunque logra superarlo probablemente exista una configuración que logre mejores resultados, dado que en esencia se está realizando la misma operación aunque sin otros operadores característicos de un AE.

E. Eficiencia Computacional

Como fue mencionado en la sección II-F el algoritmo evolutivo fue implementado permitiendo su ejecución paralela. Esto permite alcanzar una mayor capacidad de procesamiento y por consiguiente un menor tiempo de procesamiento.

El siguiente gráfico muestra como el uso de una mayor cantidad de CPUs disminuye el tiempo de ejecución :

Alg1	Alg2	pvalue
EA	local_search	2.1911×10^{-11}
EA	gaussian	4.3377×10^{-7}
local_search	gaussian	8.4377×10^{-14}

TABLE XI
P-VALORES RESULTADO DEL TEST POST HOC DE LA INSTANCIA FOX

Alg1	Alg2	pvalue
EA	local_search	2.8250×10^{-27}
EA	gaussian	0.3197
local_search	gaussian	3.9754×10^{-16}

TABLE XII
P-VALORES RESULTADO DEL TEST POST HOC DE LA INSTANCIA MONALISA

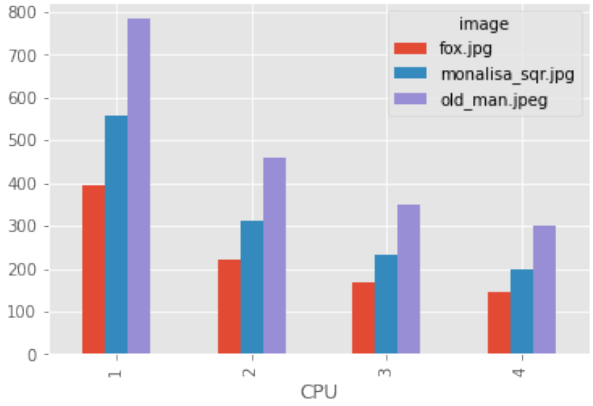


Fig. 9. Tiempo en segundos por cantidad de CPU usada.

Es clara la tendencia negativa. Sin embargo, también es claro que la disminución de tiempo requerido no es lineal a la cantidad de CPUs utilizadas lo cual se puede observar bajo la métrica *speedup* de la tabla XIV.

Valores para las otras instancias aparecen en el anexo, tablas XV y XVI.

F. Ejecuciones informales

En IV se presentarán imágenes obtenidas luego de pruebas informales del programa. De ellas se destaca la distribución no uniforme

Alg1	Alg2	pvalue
EA	local_search	3.8243×10^{-12}
EA	gaussian	0.0456
local_search	gaussian	9.7431×10^{-8}

TABLE XIII
P-VALORES RESULTADO DEL TEST POST HOC DE LA INSTANCIA OLD_MAN

CPU	time	speedup	efficiency
1	394.2663	1.0000	1.0000
2	222.0688	1.7754	0.8877
3	169.1529	2.3308	0.7769
4	145.1067	2.7171	0.6793

TABLE XIV
MÉTRICAS DESEMPEÑO EJECUCIÓN PARALELA INSTANCIA FOX.

de los triángulos, los cuales aparecen en mayor cantidad en zonas donde hay mayor variación de colores.

IV. Conclusiones

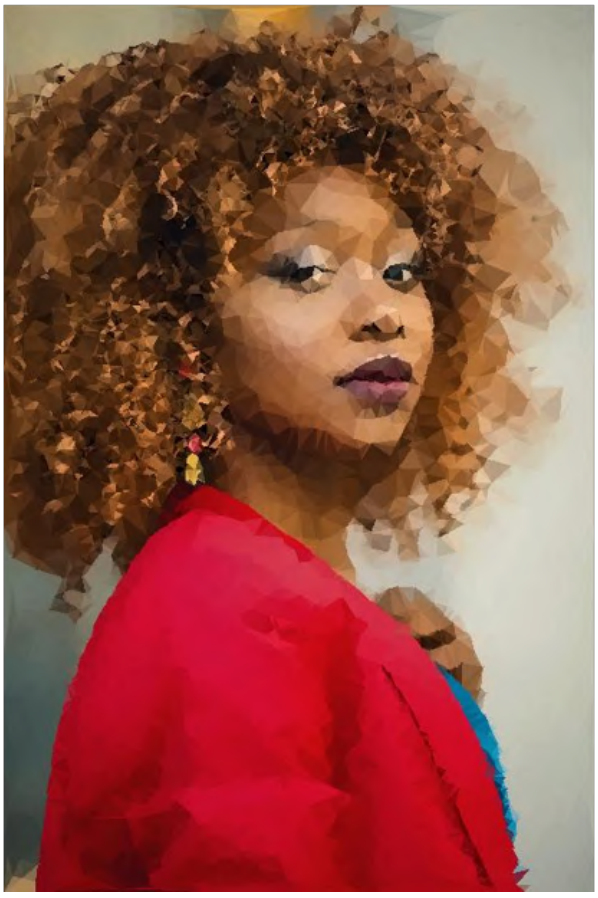
Es claro que el algoritmo planteado tiene muchos detalles a mejorar. Se resalta de él aún así características como su inicialización (mejoras como la de detección de bordes tanto como el denoising fueron detalles claves para obtener individuos más aptos e imágenes de mejor calidad), además de la velocidad con la que se logra el objetivo principal comparado con otros métodos utilizados como comparación, lo cual no se habría logrado sin el uso de múltiples procesos.

En cuanto a detalles a mejorar, la comparación con otros algoritmos lleva a la conclusión de que es probable que falte una mayor exhaustividad en cuanto a la parametrización del algoritmo. Es claro que un algoritmo evolutivo tiene una cantidad masiva de parámetros e intentar todas las combinaciones es imposible en un tiempo acotado. Es más, distintas imágenes podrían ser buenas para distintas combinaciones por lo que seguramente existan múltiples ”mejores combinaciones”, dificultando aún más el trabajo.

Es claro además que en este tipo de problemas donde se trabaja con procesamiento de imágenes lenguajes como C o C++ serían una opción indudablemente superior si de eficiencia computacional se tratase. No se considera a Python una mala elección dado que el trabajo humano se reduce ampliamente, lo cual es fundamental en situaciones tanto empresariales como universitarias donde el tiempo es crucial. Aún así, se considera beneficioso el uso de librerías como numpy o cv2, que utilizan rutinas implementadas en C y hacen su trabajo especialmente rápido aún estando en un lenguaje débilmente tipado e interpretado. Se destaca además que al utilizar múltiples cajas negras (funciones ya implementadas) tales como la librería DEAP, Delaunay, o Pillow fue necesario utilizar muchos casteos de tipos que podrían haberse evitado haciendo las implementaciones de forma manual.

En cuanto a la representación, es claro que no fue la mejor elegida dadas las características de las coordenadas. Es posible que un genotipo alternativo como una matriz booleana, donde cada valor tiene un 1 si la coordenada correspondiente es un vértice de la imagen. Esto mejoraría los problemas que se dan en el cruzamiento aunque empeoraría en cuanto a memoria utilizada. Se destaca por otro lado que los métodos utilizados logran mitigar los defectos de la representación, aunque sin duda hace falta investigar esto con más detalle.

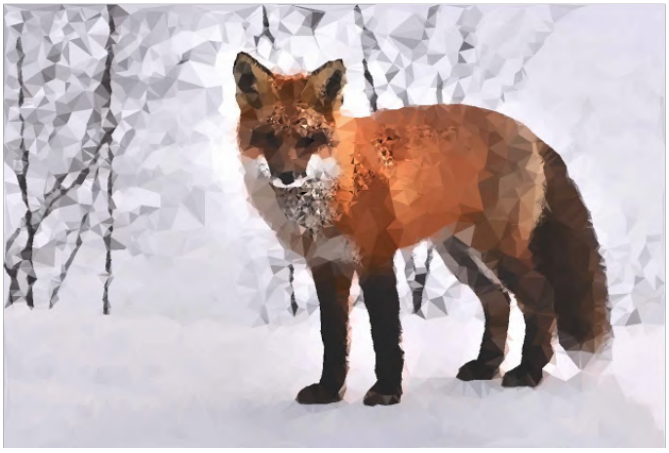
ANEXO









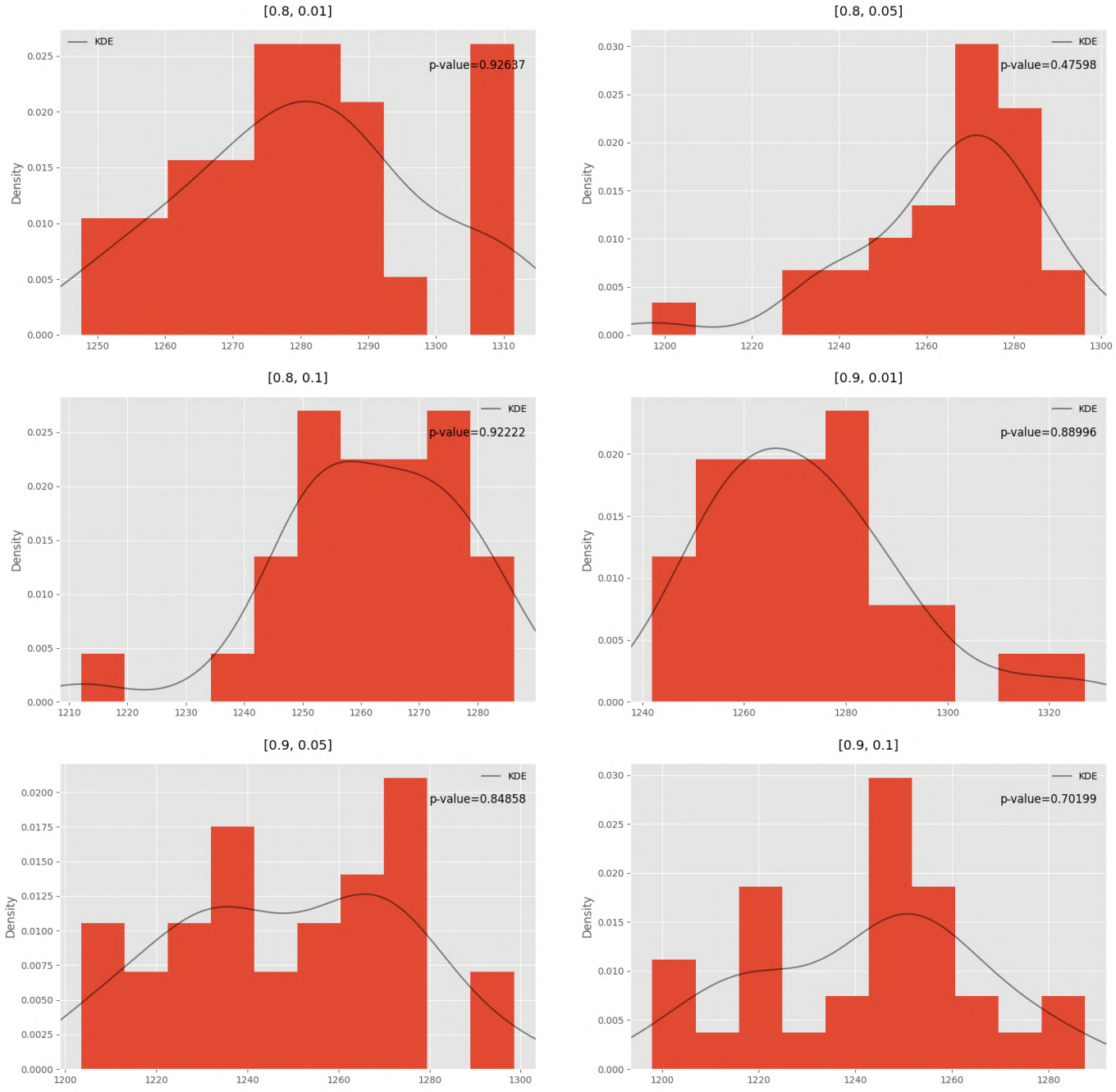


Fig. 10. Histograms and p-values associated to each configuration

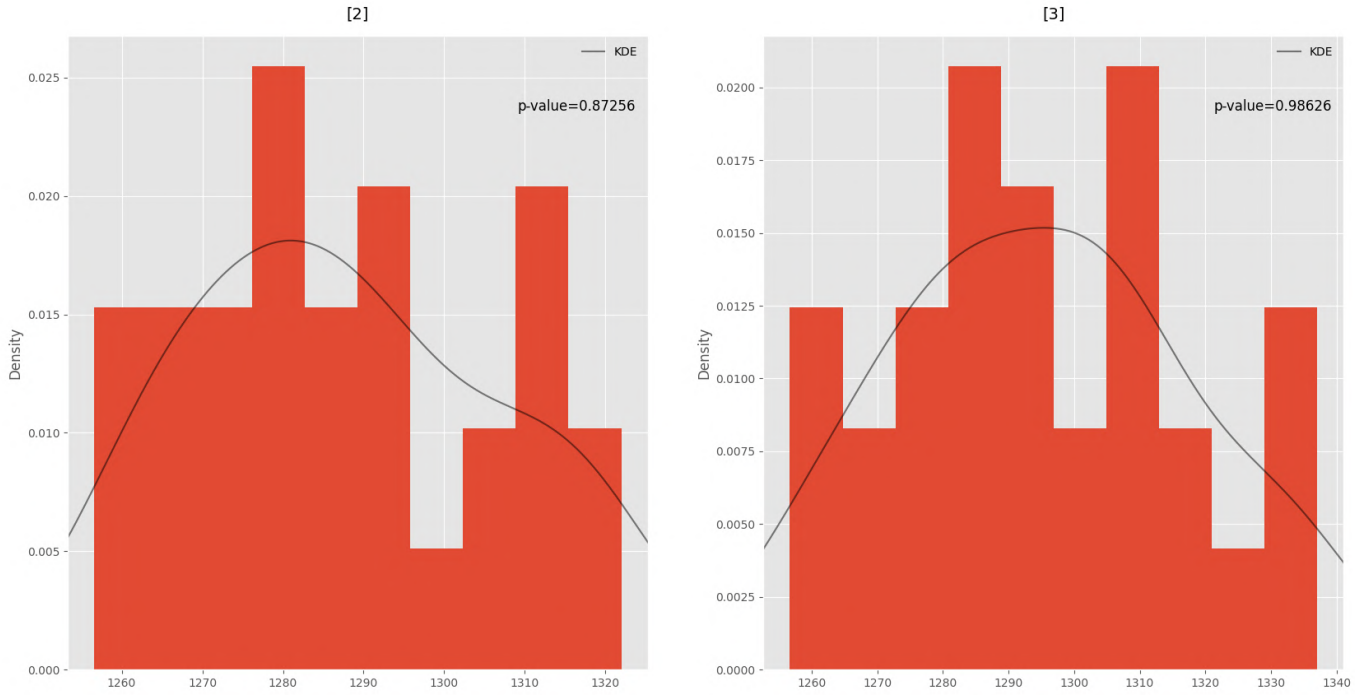


Fig. 11. Histograms and p-values associated to each $tournsize$ value

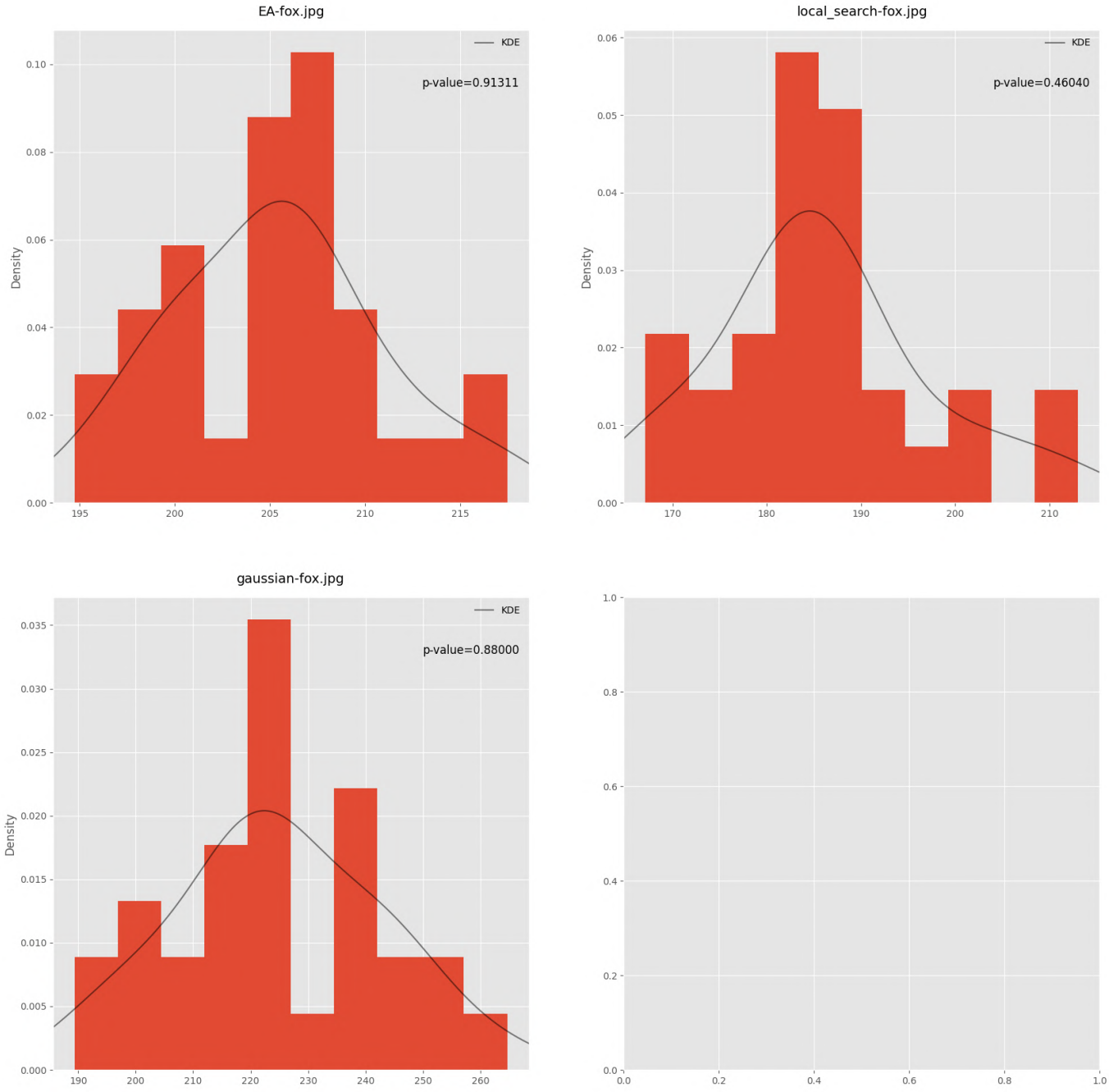
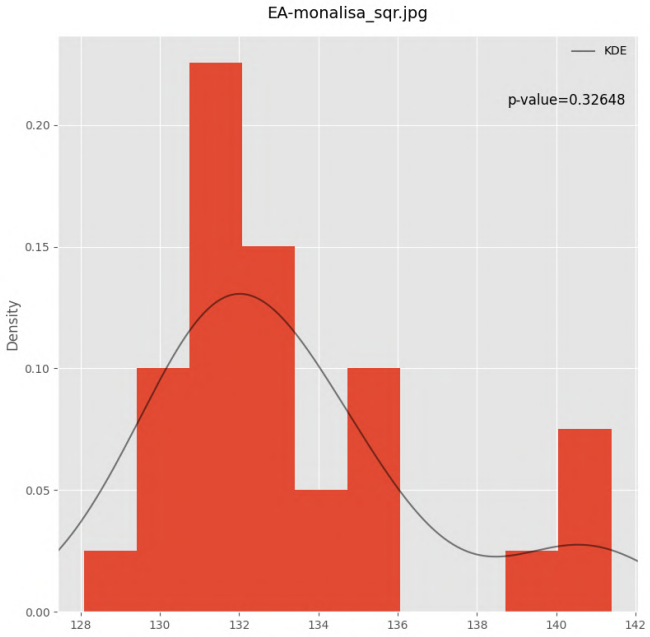
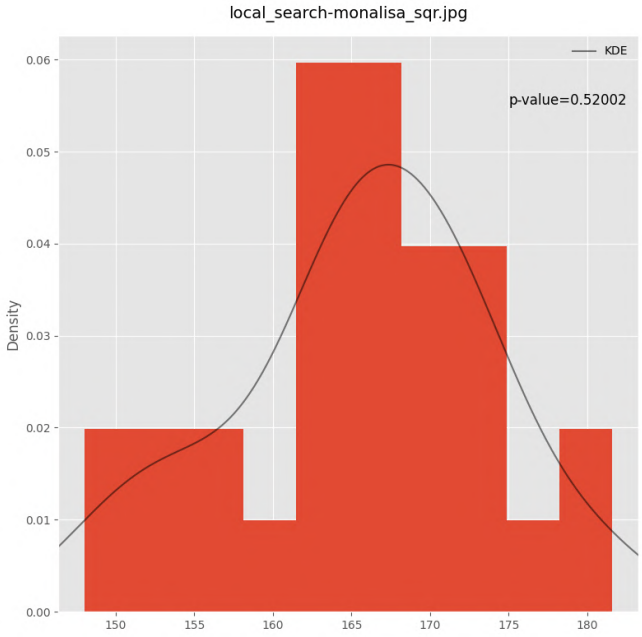
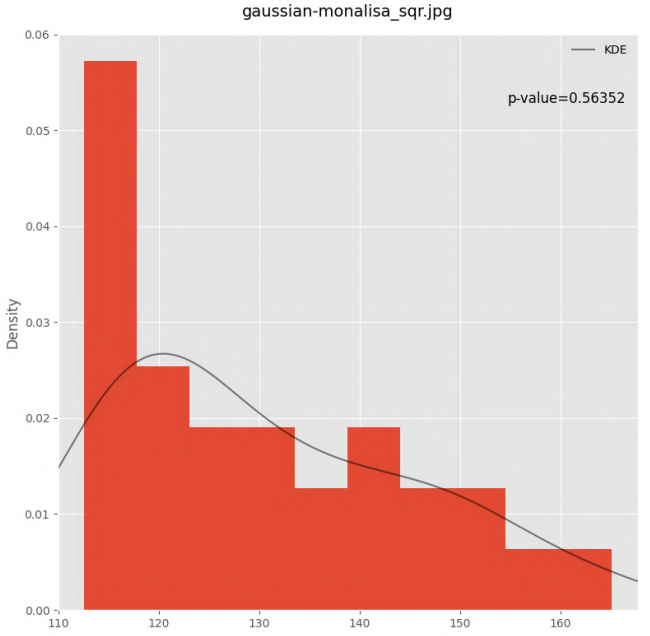
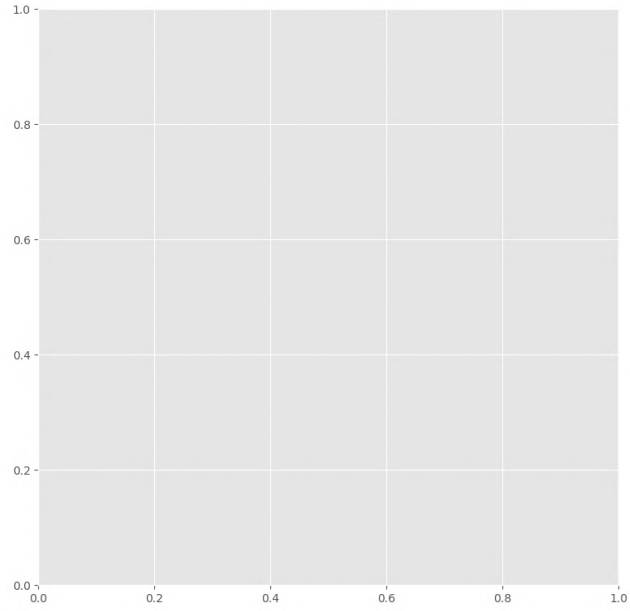


Fig. 12. Histograms and p-values associated to each algorithm for ”fox” instance


EA-monalisa_sqr.jpg
KDE
p-value=0.32648
Density
0.00
0.05
0.10
0.15
0.20
128
130
132
134
136
138
140
142


local_search-monalisa_sqr.jpg
KDE
p-value=0.52002
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
150
155
160
165
170
175
180


gaussian-monalisa_sqr.jpg
KDE
p-value=0.56352
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
110
120
130
140
150
160


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

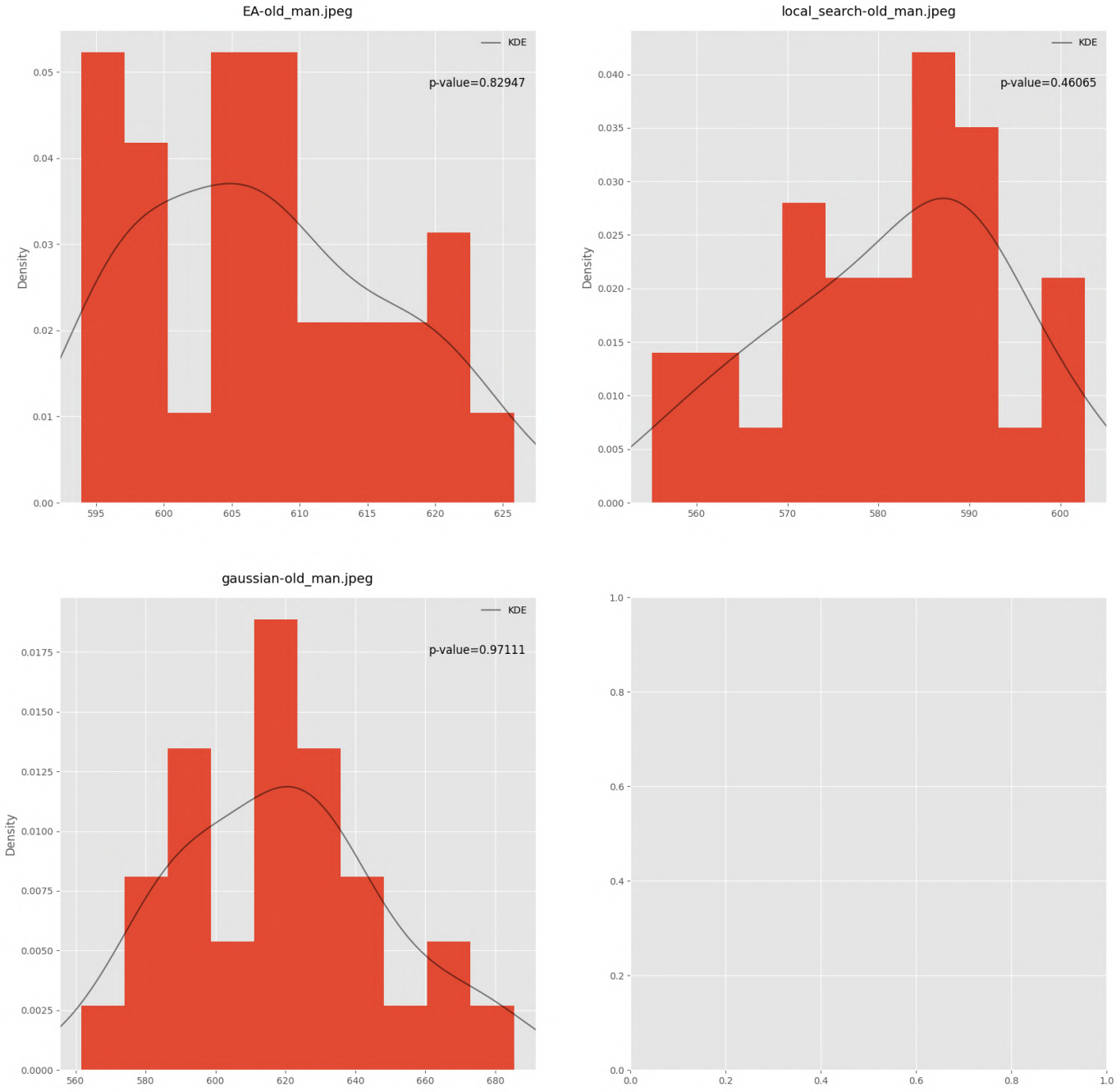


Fig. 14. Histograms and p-values associated to each algorithm for ”old_man” instance

Fig. 15. Instancia ”ultima_cena.jpg” usada en la evaluación experimental

Fig. 16. Instancia ”old_man.jpeg” usada en la evaluación experimental

Fig. 17. Instancia ”fox.jpg” usada en la evaluación experimental

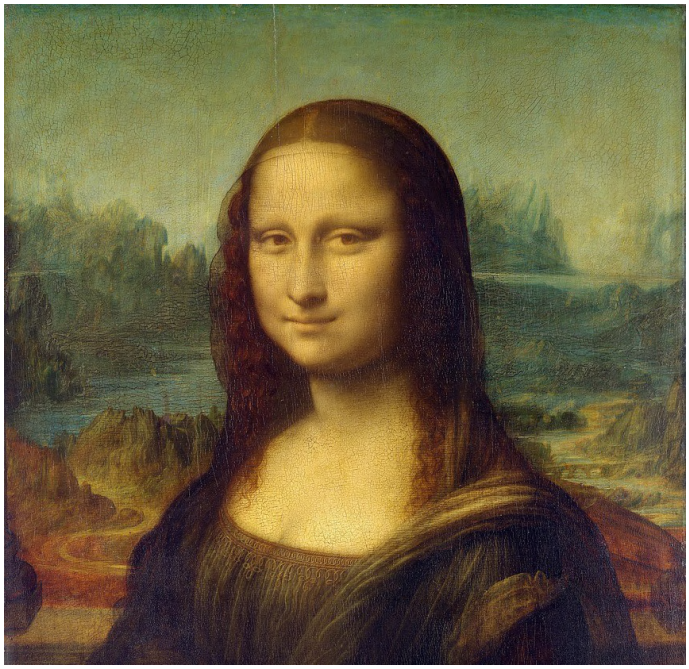


Fig. 18. Instancia ”monalisa_sqr.jpg” usada en la evaluación experimental

CPU	time	speedup	efficiency
1	557.4455	1.0000	1.0000
2	311.9573	1.2638	0.6319
3	233.5849	1.6879	0.5626
4	197.8120	1.9931	0.4983

TABLE XV
MÉTRICAS DESEMPEÑO EJECUCIÓN PARALELA INSTANCIA MONALISA_SQR.

CPU	time	speedup	efficiency
1	783.0731	1.0000	1.0000
2	460.7971	0.8556	0.4278
3	348.8206	1.1303	0.3768
4	301.5565	1.3074	0.3269

TABLE XVI
MÉTRICAS DESEMPEÑO EJECUCIÓN PARALELA INSTANCIA OLD_MAN.

0.8 0.01	0.8 0.05	0.8 0.1	0.9 0.01	0.9 0.05	0.9 0.1
1285.6718	1296.2316	1257.1360	1282.1887	1270.8018	1247.1361
1277.8005	1266.7087	1277.8595	1293.3680	1203.5224	1222.5994
1283.5207	1282.1819	1257.9150	1267.7370	1268.1959	1278.8437
1268.6046	1296.2809	1212.0996	1276.4848	1238.1162	1276.7174
1266.4327	1197.1615	1278.3394	1276.0069	1253.7022	1248.5497
1249.9504	1234.0554	1278.7676	1288.5598	1290.6218	1249.4327
1272.4721	1273.4345	1272.4503	1259.5005	1232.3408	1287.5816
1279.1580	1273.5890	1269.1835	1241.9385	1222.1688	1197.9369
1263.9115	1264.8539	1253.5634	1264.5462	1208.2918	1243.9483
1311.4938	1272.8676	1261.7821	1274.8375	1263.8410	1269.3246
1290.4645	1279.7795	1286.2260	1257.0741	1233.0469	1247.9482
1288.0625	1261.9743	1268.5354	1257.0056	1230.4345	1250.9151
1309.0145	1269.0062	1257.8357	1311.0611	1215.1523	1258.5598
1290.9403	1269.0272	1248.5782	1276.7160	1298.5209	1204.5557
1276.8797	1246.9403	1253.9363	1271.2036	1247.3562	1258.7548
1308.3408	1259.4911	1281.7611	1250.6130	1269.0497	1244.8141
1284.3283	1253.7797	1274.9837	1268.2817	1275.5456	1267.3656
1291.1544	1268.7488	1253.9245	1265.6365	1208.7293	1220.5942
1278.5617	1284.8983	1273.1054	1284.3675	1270.2110	1236.4464
1294.3869	1268.3956	1286.0519	1295.0319	1278.8329	1255.8330
1274.3965	1251.7734	1249.3824	1288.0501	1273.7892	1218.3705
1270.1234	1284.7593	1250.7912	1327.0271	1234.0046	1248.7563
1247.6159	1237.5675	1263.8418	1257.1326	1226.1833	1214.4975
1257.3569	1269.9647	1252.7585	1249.3579	1260.1345	1216.1588
1262.4379	1262.7887	1244.8466	1254.7337	1262.8462	1259.9421
1306.1583	1281.6706	1268.4594	1262.9895	1235.4076	1218.8816
1280.4407	1279.9461	1248.0241	1278.2913	1255.7811	1227.7410
1285.4793	1238.4895	1269.8589	1259.6509	1272.6496	1206.3900
1306.4801	1277.4858	1264.4755	1269.5254	1230.9654	1256.9984
1255.7520	1236.3769	1236.7613	1247.8087	1243.5220	1241.1383

TABLE XVII
RESULTADOS NUMÉRICOS DEL MEJOR FITNESS OBTENIDO EN CADA EJECUCIÓN PARA CADA CONFIGURACIÓN PARAMÉTRICA

2	3
1295.0564	1332.4642
1314.0835	1307.8886
1321.9536	1319.9888
1280.4055	1286.9034
1315.2798	1268.5276
1293.7734	1301.6071
1277.6218	1256.6834
1262.9631	1272.2587
1320.7925	1273.5196
1304.8608	1304.5471
1286.4181	1307.0920
1261.7812	1261.2181
1302.9913	1329.4915
1267.1699	1286.1576
1264.3638	1277.8901
1280.9976	1290.5736
1289.6851	1282.7918
1312.3864	1282.0402
1286.9146	1283.5717
1272.8391	1312.9531
1290.5929	1260.1048
1311.0519	1328.9160
1289.0652	1295.6980
1256.5770	1278.5302
1264.9878	1296.0841
1275.3501	1306.5349
1279.1265	1308.4746
1273.2254	1305.8820
1297.6123	1296.5932
1277.8450	1337.0260

TABLE XVIII
RESULTADOS NUMÉRICOS DEL MEJOR FITNESS OBTENIDO EN CADA EJECUCIÓN PARA LOS DIFERENTES VALORES DE $tournsize$

EA-fox	ls-fox	g-fox	EA-monalisa	ls-monalisa	g-monalisa	EA-old_man	ls-old_man	g-old_man
206.2058	178.7275	217.9447	128.0883	170.9017	121.6847	620.1102	592.3749	561.6418
204.5051	213.0275	251.5757	141.3873	164.2051	126.9666	625.7992	557.0035	581.0144
202.4320	180.8425	204.3949	129.6351	166.8769	116.5370	601.6035	570.5601	620.2943
207.0510	188.7812	198.8661	135.2533	169.8875	112.5757	608.2018	602.7074	646.8996
199.2354	187.0419	244.7806	131.0436	165.7596	139.9465	619.9887	587.6206	632.2120
207.0242	202.2994	240.3157	130.0731	164.0303	165.0247	605.3587	587.5126	657.6629
207.4972	181.4400	221.3665	129.9022	150.6502	120.9623	599.2735	563.1057	591.9690
209.0009	188.7611	202.8646	139.0126	167.9505	122.6007	595.2606	588.2255	591.3814
204.7178	184.3599	230.8892	134.9930	163.1332	136.1244	617.7540	595.4134	590.8487
212.7071	181.1839	237.6027	140.9338	181.5786	117.7051	612.1374	583.1280	633.5005
204.3450	209.2648	223.4898	140.7503	152.1391	148.0525	609.1867	591.1248	612.0691
194.7450	186.6593	234.6259	131.6095	160.0193	143.5785	603.4724	601.5338	627.1506
196.4175	173.3540	194.0197	131.5338	164.5477	120.1403	619.0388	580.2602	587.7564
205.3671	188.3459	215.6567	132.8496	167.2136	116.2292	607.0347	599.5173	671.7710
199.2126	167.1225	226.3300	133.3322	175.3009	117.3582	599.1969	563.2907	618.5382
201.1157	167.3361	218.4415	131.8696	179.2272	125.2805	596.4124	583.9168	579.4784
209.9922	177.8215	208.0153	129.4282	156.4704	152.0982	615.3960	589.3143	672.5558
208.2508	193.3161	225.4000	131.4776	172.2621	149.2224	605.8951	571.3749	595.1651
198.5744	189.2554	251.9157	134.2414	173.3159	116.6437	607.6637	571.9280	622.2419
206.3213	183.2192	225.9861	131.5077	156.6312	128.3394	597.2828	592.0346	611.5013
204.5470	167.9364	216.2909	133.5893	173.3001	131.9909	622.0458	565.0941	619.1463
199.8934	182.0464	238.8941	131.5835	167.7425	158.4678	598.0827	587.7453	605.4309
205.0182	184.3949	223.2209	132.2594	170.9937	150.2719	614.1677	577.9632	581.3623
213.0499	198.5511	264.6492	131.1639	172.4809	135.5998	593.9021	587.9500	647.7596
200.6789	181.2217	240.5804	131.2341	148.0444	112.8297	595.6968	578.7406	630.9273
210.5742	186.5493	211.2079	133.0896	163.1085	124.2936	611.3639	570.2534	631.3304
200.9346	192.9215	220.0357	135.3204	165.7942	132.2776	604.9042	576.8815	619.9482
216.1583	185.2223	247.2411	135.9135	169.4846	141.3947	607.7598	555.1271	602.0262
217.4636	203.0170	189.3914	133.2931	153.6176	117.7744	596.8963	590.5425	685.3580
206.1149	173.4219	221.8556	133.1114	163.2394	114.0330	604.0266	579.1382	638.4350

TABLE XIX
RESULTADOS NUMÉRICOS DEL MEJOR FITNESS OBTENIDO EN CADA EJECUCIÓN PARA CADA PAR ALGORITMO-INSTANCIA

REFERENCES

[1] Delaunay, Boris (1934). "Sur la sphère vide". Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles. 6: 793–800.

[2] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, Non-Local Means Denoising, Image Processing On Line, 1 (2011), pp. 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm

[3] OpenCV-Python. https://docs.opencv.org/

[4] Li, Q., Wang, B., Fan, S. (2009). Browse Conference Publications Computer Science and Engineer ... Help Working with Abstracts An Improved CANNY Edge Detection Algorithm. In 2009 Second International Workshop on Computer Science and Engineering proceedings : WCSE 2009 : 28–30 October 2009, Qingdao, China (pp. 497–500). Los Alamitos, CA: IEEE Computer Society

[5] Kolmogorov-Smirnov test for goodness of fit. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html

[6] ANOVA scipy library implementation. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

[7] Pairwise T test for multiple comparisons of independent groups. https://scikit-posthocs.readthedocs.io/en/latest/generated/scikit_posthocs.posthoc_ttest/

[8] Holm–Bonferroni method https://en.wikipedia.org/wiki/Holm%E2%80%93Bonferroni_method

[9] Family-wise error rate https://en.wikipedia.org/wiki/Family-wise_error_rate