“Generación de Pinturas Utilizando la Técnica Artística Puntillista con Algoritmos Genéticos”

Autor: David Fosca Gamarra

Resumen - Generar pinturas en base a algoritmos computacionales es un tema que se ha tratado en muchas ocasiones. Sin embargo, no muchos trabajos han presentado a detalle cómo es que esta actividad puede ser abordada por algoritmos genéticos y cómo estos pueden ser aplicados en el ámbito del arte. Cabe resaltar que en este ámbito subjetivo es difícil determinar lo que se puede considerar como una solución óptima y que obtener como resultado exactamente la misma imágen no es lo que se desea, por lo que la mejora del fitness servirá netamente como heurística y no como el fin absoluto. El objetivo es analizar cómo un algoritmo genético puede replicar una imagen utilizando la técnica artística del puntillismo y crear en el proceso, lo que se puede considerar como su propia versión artística de la imagen original. Con este fin, se comparó el desempeño de utilizar la combinación de diversos operadores genéticos ajustando estratégicamente sus hiper-parámetros para obtener un mejor resultado. Se encontró que trabajando en conjunto con operadores de cruzamiento y mutación se obtienen mejores resultados que trabajando con cada uno de estos de manera individual. Así mismo, para este problema en específico, es beneficioso tener una probabilidad de mutación del 100% y disponer de la posibilidad de aumentar genes al individuo.

1. Introducción.

- A lo largo de la historia de la computación, muchos algoritmos y metodologías se han creado para generar expresiones artísticas como lo son las pinturas. Inclusive con el objetivo de explorar la posibilidad de dotar de “creatividad” a un sistema. Ejemplos de estos esfuerzos son el trabajo de Y. Zhao **[1]** o el de A. Kasao **[2]** que se basan en el uso de redes neuronales. Dentro del campo de la Inteligencia Artificial, existe lo que se conoce como Algoritmos Genéticos, cuyas aplicaciones son muy variadas por su flexibilidad de uso y capacidad de búsqueda de soluciones óptimas en la industria. Así mismo, tienen un potencial interesante para ser aplicados en el ámbito de las artes y lograr inclusive que un sistema computarizado pueda crear su propia versión artística de una fotografía. El trabajo desarrollado en el presente artículo, está inspirado en trabajos como el de A. Hansen y C. Lewis **[3]** que pretende replicar fotografías utilizando trazos, variando sus posiciones, colores y tonalidades; o el de R. Johansson **[4]** que propone una manera interesante de replicar la “Mona Lisa”, utilizando polígonos como base de representación artística. Ambos trabajos utilizan unidades básicas para la construcción de la pintura, ya sea trazos rectangulares o poligonales.
- El objetivo del trabajo es demostrar de forma práctica cómo los algoritmos genéticos pueden ser utilizados para generar una pintura utilizando círculos como unidad básica de construcción, asemejándose al estilo artístico del puntillismo **[5]** y teniendo a una imágen como referencia para replicarla.
- Así mismo, se desean identificar las consideraciones y características más importantes que un algoritmo genético debería tener para llegar a un resultado que realmente represente a la imagen de referencia y sea visualmente atractivo.

2. Metodología.

- **La Técnica de Puntillismo:** Para reproducir este estilo artístico, se trabajó utilizando como herramienta de representación únicamente círculos. De los cuales se podrán modificar las siguientes características durante el proceso de búsqueda: i) Color (r,g,b), ii) Posición (x,y) y iii) Diámetro.
- **El Algoritmo Genético (AG):** Para obtener la mejor combinación de las características previamente mencionadas, se implementó un AG, donde cada gen representa estas características. Se desarrollaron cuatro tipos diferentes de operadores de cruzamiento (“OnePoint”, “Uniform”, “Arithmetic-Simple” y “Arithmetic-Complete”) y dos tipos de operadores de mutación (“Single-Gene” y “All-Genes”) con el objetivo de experimentar con ellos y determinar qué combinación es la más óptima para la reproducción de una imágen. Así mismo, a modo de prueba final, se probaron dos tipos de operadores de selección de padres (“Tournament” u “Roulette”). A continuación se explica cómo funciona cada uno de estos operadores en el contexto del problema.
  1. Cruzamiento - OnePoint.
     Una vez seleccionados dos padres para el proceso de cruzamiento, se escoge un punto de corte de manera aleatoria en base al tamaño de uno de los dos cromosomas padres, y se intercambian sus genes. Se determinó que en este proceso se intercambiarán los genes completos, es decir, se transmite la información completa del círculo (Color, Posición y Diámetro) para preservarla.
     Consideración: Si los cromosomas son de diferentes tamaños y el punto de corte se encuentra fuera del tamaño de uno de estos, entonces el punto de corte es igual al menor tamaño del cromosoma.
  2. Cruzamiento - Uniform.
     Una vez seleccionados dos padres para el proceso de cruzamiento, se escogen de manera aleatoria los genes que serán intercambiados entre ambos padres para generar nuevos hijos. Al igual que el operador anterior, se determinó que en este proceso se intercambiarán los genes completos, ya que un gen contiene 3 parámetros diferentes y para seleccionar alguno de estos se tendría que hacer mediante un proceso estocástico, el cual puede interferir con la explotación de resultados de los mejores individuos.
     Consideración: Si los cromosomas son de diferentes tamaños, entonces solo se podrán intercambiar los genes disponibles hasta el tamaño del menor cromosoma.
  3. Cruzamiento - Arithmetic (Simple).
     Una vez seleccionados dos padres, se genera el nuevo cromosoma aplicando la *Fórmula 1* sobre las tres características (color, posición y diámetro) de los genes de los padres. La posición del gen_hijoN se escoge de manera aleatoria y corresponde a ambos padres. El valor *alpha* es un número aleatorio entre 0 y 1.
     Fórmula 1:
     *gen_hijo1 = alpha *gen_padre1 + (1−alpha) *gen_padre2*
     *gen_hijo2 = alpha *gen_padre2 + (1−alpha) *gen_padre1*
     Consideración: Si los cromosomas son de diferentes tamaños y la posición del gen aleatorio se encuentra fuera del tamaño de uno de estos, entonces la posición del gen aleatorio es igual al tamaño del menor cromosoma.
  4. Cruzamiento - Arithmetic (Complete).
     Una vez seleccionados dos padres para el proceso de cruzamiento, se genera el nuevo cromosoma aplicando la *Fórmula 2* sobre las tres características (Color, Posición y Diámetro) de cada uno de los genes de ambos padres. El valor *alpha* es un número aleatorio entre 0 y 1.
     Fórmula 2:
     *hijo1 = alpha *padre1 + (1−alpha) *padre2*
     *hijo2 = alpha *padre2 + (1−alpha) *padre1*
     Consideración: Si los cromosomas son de diferentes tamaños entonces solo se podrá considerar para la fórmula los genes disponibles hasta el tamaño del menor cromosoma, el resto de genes que no tienen una pareja se pasan iguales.
  5. Mutación - SingleGene.
     Una vez se tenga la nueva población, se determina de manera aleatoria en base al hiper-parámetro *pmut*, cuántos cromosomas sufrirán una mutación en sus genes. En caso de que un cromosoma sea seleccionado, se escogerá de manera

aleatoria lo siguiente: i) Gen a modificar y ii) Característica del gen (Color, Posición y Diámetro) que se mutará. Dependiendo de las características, se aplica la *Fórmula 3* sobre el gen. El valor *beta* es un número aleatorio entre -1 y 1 y el hiper-parámetro *pint* define la intensidad de la mutación.

Fórmula 3:

*color = color + pint * beta *color*
*posición = posición + pint *beta *posición*
*diámetro= diámetro + pint * beta *diámetro*

Al igual que en el proceso artístico del puntillismo se pueden ir agregando nuevos círculos sobre la imágen, el algoritmo de mutación también permite realizar este proceso agregando un nuevo gen en caso el hiper-parámetro *paddgen* sea mayor a un valor aleatorio entre 0 y 1.

6. Mutación - AllGenes.

   Una vez se tenga la nueva población, se determina de manera aleatoria en base al hiper-parámetro *pmut*, cuántos cromosomas sufrirán una mutación en sus genes. En caso de que un cromosoma sea seleccionado, se escogerá de manera aleatoria la característica del gen (Color, Posición y Diámetro) que se mutará. Dependiendo de las características, se aplicará la *Fórmula 3* sobre todos sus genes.

   Al igual que el operador anterior, también permite agregar un nuevo gen en caso el hiper-parámetro *paddgen* sea mayor a un valor aleatorio entre 0 y 1.

3. Experimentación y Resultados.

- **Herramientas de Diseño e Implementación:** El algoritmo genético fue diseñado e implementado en Python 3.7 y se utilizó la distribución de Anaconda para trabajar la ejecución de manera local (computadora personal). A continuación se presentan las características del equipo donde se realizaron las pruebas. Marca: ASUS, RAM : 12GB , Procesador: Intel Core i7 CPU 2.4 GHz, SO: Windows 64 bits.
- **Datos de Entrada:** Debido a las características de este proyecto, los datos de entrada para el algoritmo son: i) imágen de referencia para realizar la reproducción con puntillismo, e ii) hiper-parámetros propios del algoritmo genético que deben ser ajustados para mejorar su desempeño. Cabe resaltar que para las pruebas iniciales se utilizó una única imágen de referencia con formas y colores muy básicos para aumentar la posibilidad de encontrar un óptimo global en menor tiempo.
- **Métrica de Evaluación:** La principal métrica para evaluar el desempeño del AG es el "fitness". Calculado con la *Fórmula 4***,** donde un fitness cercano o igual a 100 implica que la diferencia entre la imagen original (*AG_origin*) y la generada por el AG (*AG_img*) tienda a ser cero. Por lo cual, el algoritmo busca maximizar el fitness y en consecuencia disminuir la diferencia entre las imágenes. Por otro lado, se considera el tiempo de ejecución como un factor a tomar en cuenta, ya que puede ser decisivo dependiendo de la situación en la que se aplique el AG.

  Fórmula 4:

  fitness = 100 / (1 + abs(AG_img - AG_origin))
- **Experimentos Realizados:** Con el AG implementado, se realizaron diversas pruebas divididas en dos bloques. El Bloque 1, tiene como objetivo encontrar la mejor combinación de operadores de cruzamiento, mutación y selección de padres, así como los hiper-parámetros correspondientes, mediante la combinación estratégica entre estos. En el Bloque 1 se trabajó con una población de 50 individuos con una cantidad inicial de 100 genes cada uno y un número limitado de generaciones (500) debido al mayor tiempo que implicaría ejecutar un mayor número de generaciones de forma reiterativa (10 veces) para obtener resultados estadísticamente válidos. En el Bloque II, se tomó como base lo aprendido en el Bloque I y se ejecutó el AG con la mejor combinación de operadores e hiper-parámetros encontrados, sin embargo, se trabajó con una mayor cantidad de generaciones (>10,000) y diferentes imágenes para visualizar los resultados sobre la imagen reproducida.

Bloque I

Se seleccionó una imagen "simple" para las pruebas. La elección se hizo en base a: i) Poca variedad de colores, ii) Bajo nivel de detalle. Para el AG implementado, es necesario modificar el tamaño de la imagen a 200x200. Se detallan las pruebas a continuación.

1. Solo Operadores de Cruzamiento.
   - OnePoint.
   - Uniform.
   - Arithmetic-Simple
   - Arithmetic-Complete
2. Solo Operadores de Mutación.
   - SingleGene.
   - AllGenes.
3. Operadores de Cruzamiento y Mutación (paddgene=0.1, imut=0.5).
   - OnePoint + SingleGene (pmut:0.5)
   - OnePoint + SingleGene (pmut:0.7)
   - OnePoint + SingleGene (pmut:1.0)
   - OnePoint + AllGenes (pmut:0.5)
   - OnePoint + AllGenes (pmut:1.0)
   - Uniform + SingleGene (pmut:0.5)
   - Uniform + SingleGene (pmut:0.7)
   - Uniform + SingleGene (pmut:1.0)
   - Uniform + AllGenes (pmut:0.5)
   - Uniform + AllGenes (pmut:1.0)
   - Arithmetic-Complete + SingleGene (pmut:0.5)
   - Arithmetic-Complete + SingleGene (pmut:1.0)
   - Arithmetic-Complete + AllGenes (pmut:0.5)
   - Arithmetic-Complete + AllGenes (pmut:1.0)
   - Arithmetic-Simple + SingleGene (pmut:0.5)
   - Arithmetic-Simple + SingleGene (pmut:1.0)
   - Arithmetic-Simple + AllGenes (pmut:0.5)
   - Arithmetic-Simple + AllGenes (pmut:1.0)

Bloque II.

- Uniform + SingleGene (pmut:1.0, paddgene:0.1, imut:0.5, gen:2000)
- Uniform + SingleGene (pmut:1.0, paddgene:0.2, imut:0.7, gen:2000)
- Uniform + SingleGene (pmut:1.0, paddgene:0.5, imut:0.7, gen:2000)
- Uniform + SingleGene (pmut:1.0, paddgene:0.2, imut:1.0, gen:2000)
- Uniform + SingleGene (pmut:1.0,paddgene:0.2, imut:0.7, gen:10000)
- Uniform + SingleGene (pmut:1.0,paddgene:0.2, imut:0.7, gen:15000)

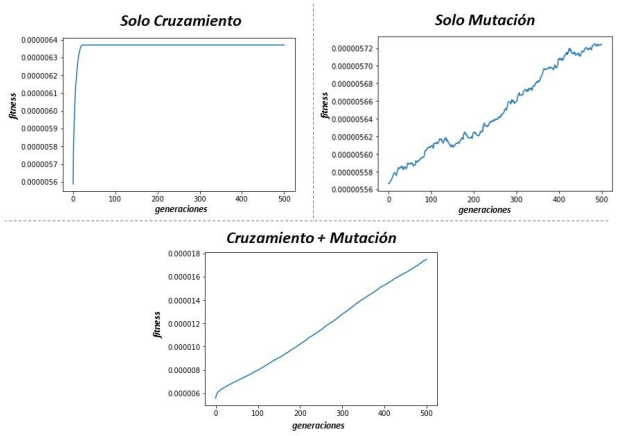
- **Resultados y Discusión:** Los resultados correspondientes a las pruebas realizadas en el Bloque I se presentan en la *Tabla 1*.

Tabla 1: *Resultados del Bloque I.*

Población: 50 | Tamaño de Cromosoma Inicial: 100 | Número de Generaciones: 500 | Selección de Padres: Tournament									
Operadores e Hiper-Parámetros del AG					Resultados de Ejecución (10 veces)				
Cruzamiento	Mutación	pmut	imut	padd	Fitness (Media)	Fitness (STD)	Tiempo (min)	Mejor Cromosoma (# genes)	¿Convergencia Precoz?
None	Single-Gene	0.8	0.5	0.1	5.72E-06	4.03E-08	3.3	136	No
		0.8	0.8	0.5	6.39E-06	5.48E-08	4.3	281	No
	All-Genes	0.8	0.5	0.1	5.07E-06	2.20E-08	4.6	140	Si
		0.8	0.8	0.5	5.10E-06	2.77E-08	5.4	160	Si
OnePoint	None	0.0	0.0	0.0	6.37E-06	8.05E-08	0.5	100	Si
	Single-Gene	0.5	0.5	0.1	1.75E-05	6.87E-07	1.8	149	No
		0.7	0.5	0.1	1.89E-05	1.55E-06	2.5	147	No
		1.0	0.5	0.1	1.99E-05	4.68E-07	3.6	154	No
	All-Genes	0.5	0.5	0.1	1.43E-05	1.06E-06	2.4	101	Si
		1.0	0.5	0.1	1.49E-05	4.52E-07	2.8	103	Si
Uniform	None	0.0	0.0	0.0	7.11E-06	9.24E-08	3.3	100	Si
	Single-Gene	0.5	0.5	0.1	2.00E-05	9.78E-07	5.1	140	No
		0.7	0.5	0.1	2.16E-05	5.06E-07	6	138	No
		1.0	0.5	0.1	2.31E-05	7.97E-07	7	146	No
	All-Genes	0.5	0.5	0.1	1.43E-05	1.26E-06	4.6	101	Si
		1.0	0.5	0.1	1.53E-05	2.80E-07	5.9	103	Si
Arithmetic-Simple	None	0.0	0.0	0.0	6.05E-06	4.44E-08	3.7	100	Si
	Single-Gene	0.5	0.5	0.1	9.28E-06	1.41E-07	5.6	215	No
		1.0	0.5	0.1	1.02E-05	1.47E-07	6.7	235	No
	All-Genes	0.5	0.5	0.1	9.06E-06	2.98E-06	5.1	152	No
		1.0	0.5	0.1	1.42E-05	2.52E-06	5.6	113	No
Arithmetic-Complete	None	0.0	0.0	0.0	6.04E-06	3.79E-08	4.8	100	Si
	Single-Gene	0.5	0.5	0.1	1.16E-05	6.94E-07	9.4	177	No
		1.0	0.5	0.1	9.94E-06	1.72E-07	10.2	247	No
	All-Genes	0.5	0.5	0.1	1.16E-05	1.31E-06	5.6	136	No
		1.0	0.5	0.1	1.36E-05	1.31E-06	6.7	125	No

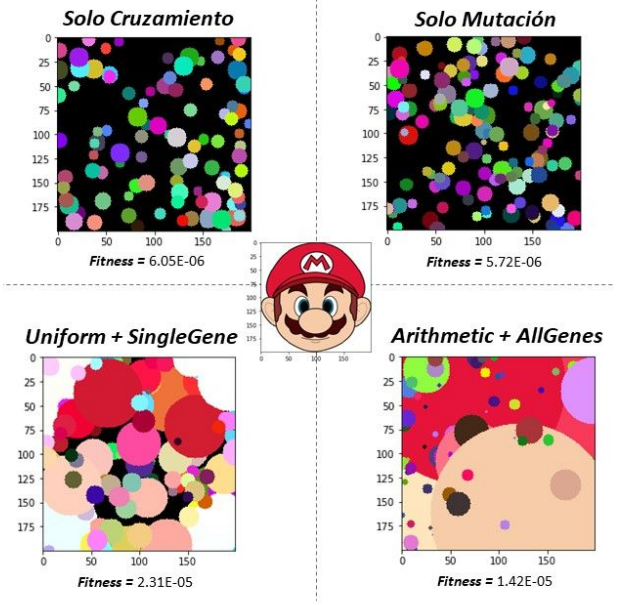
Podemos observar que trabajando únicamente con cualquiera de los cuatro operadores de cruzamiento (pmut=0), el fitness no mejora pasadas las primeras 50 generaciones (*Figura 1.a*), este fenómeno lo hemos denominado como "convergencia precoz" en la *Tabla 1*. Por otro lado, cuando se trabaja únicamente con operadores de mutación se observa una curva mucho más irregular pero cuyo fitness tiende a incrementar conforme van pasando las generaciones (*Figura 1.b*). Finalmente, cuando se trabaja con ambos operadores (mutación y cruzamiento), dependiendo de la combinación de hiper-parámetros, la mayoría de curvas de evolución son regulares y con tendencia a seguir incrementando su fitness (*Figura 1.c*).

Figura 1: *Muestra las gráfica de evolución de fitness vs generación. 1.a) implementando solamente operadores de cruzamiento, 1.b) implementando solamente operadores de mutación y 1.c) implementando ambos operadores.*



De los resultados presentados, resolver este problema únicamente con operadores de cruzamiento no es una buena opción por el bajo fitness que se obtiene (*Figura 2.a*). Por otro lado, trabajar únicamente con operadores de mutación resulta en una evolución que podría converger después de varias generaciones debido a la irregularidad de la evolución del fitness, consecuencia del proceso de únicamente explorar nuevas soluciones y no explotar las mejores ya encontradas. Esta irregularidad se corrige utilizando ambos operadores juntos. Después de 500 generaciones, vemos como el fitness de un AG con cruzamiento y mutación (*Figura 2.c y 2.d*) supera a uno trabajando solo con operaciones de mutación (Figura 2.b).

Figura 2: *Muestra las imágenes reproducidas después de 500 generaciones. 2.a) implementando solamente operadores de cruzamiento, 2.b) implementando solamente operadores de mutación, 2.c) implementando cruzamiento "Uniform" y mutación "SingleGene" y 2.d) implementando cruzamiento "Arithmetic-Simple" y mutación "AllGenes".*



De la *Tabla 1,* se aprecia que los operadores de cruzamiento "OnePoint" y "Uniform" logran sus mejores resultados de fitness trabajando con el operador de mutación "Single-Gene" con una probabilidad de mutación (pmut) igual 1.0. Por otro lado, los operadores de cruzamiento "Arithmetic-Simple" y "Arithmetic-Complete" alcanzan sus mejores valores de fitness trabajando con el operador de mutación "All-Genes" con una probabilidad de mutación (pmut) igual 1.0. En ambos grupos, una probabilidad (pmut) mayor ha generado un mejor desempeño del AG. Sin embargo, a diferencia de la mutación "Single-Gene", se observa que "All-Genes" puede generar, con operadores de cruzamiento "OnePoint" y "Uniform", que la solución tienda a converger de manera precoz y por lo tanto demore más en su búsqueda de un mejor fitness (o incluso que caiga en un óptimo local). Esto puede ser consecuencia de la manera tan "brusca" con la que este operador realiza la mutación, ya que a diferencia de "Single-Gene", "All-Genes" modifica todos los genes. Este fenómeno también lo podemos ver en la *Figura 2.d*, dónde hay genes que han sido drasticamente modificados como los genes de color rojo y rosado en ambas esquinas de la imagen. Sin embargo, en la *Figura 2.c* vemos como las pequeñas modificaciones del operador de mutación Single-Gene permiten un ajuste más preciso y que es coherente con el nivel de detalle de las imágenes.

Si bien todos los operadores de cruzamiento se benefician de la exploración brindada por la mutación, la combinación de "Uniform" con "Single-Gene" con una probabilidad (pmut) de mutación igual a 1.0, es la que mejores resultados ha presentado (fitness=2.31E-05). Así mismo, su tiempo de ejecución fue de 7 minutos (tiempo promedio de todos las pruebas: 4.8 min, tiempo máximo: 10.2 min) y el tamaño del mejor cromosoma o cantidad de círculos que se utilizaron para reproducir la imagen es de: 146 genes (tamaño promedio de todos las pruebas: 146.1 genes, tamaño máximo: 281 genes). La imagen reproducida se puede apreciar en la *Figura 2.c.*

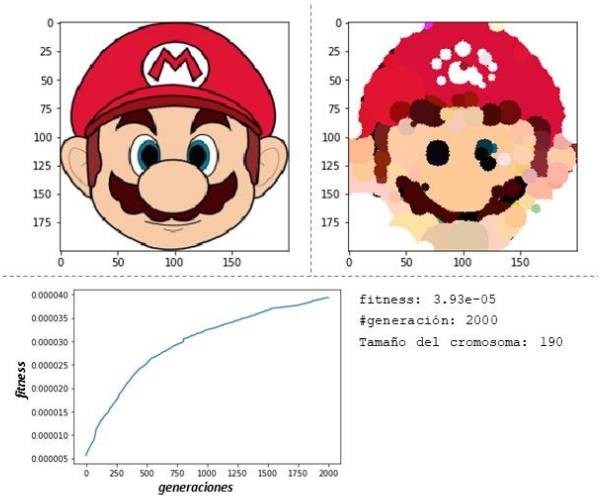
Con la mejor combinación de operadores encontrados y sus respectivos hiper-parámetros ajustados, pasamos a realizar las pruebas del Bloque II. Los resultados se muestran en la *Tabla 2.*

Tabla 2: *Resultados del Bloque II.*

Población: 50 | Tamaño de Cromosoma Inicial: 100 | Número de Generaciones: 2000| Selección de Padres: Tournament						
Operadores e Hiper-Parámetros del AG					**Resultados de Ejecución (1 vez)**	
Cruzamiento	Mutación	pmut	imut	padd	Fitness	# Genes
Uniform	**Single-Gene**	1.0	0.5	0.1	5.59E-06	160
		1.0	0.7	0.2	5.62E-06	190
		1.0	0.7	0.5	5.56E-06	293
		1.0	1	0.2	5.58E-06	174

De los resultados de la *Tabla 2*, podemos observar que el mejor fitness se obtiene con una intensidad de mutación (*imut*) de 0.7 y una probabilidad de agregar nuevos genes al cromosoma (*padd*) de 0.2. Por otro lado, cuando se incrementa *imut* o *padd* el fitness disminuye. Además, manteniendo un valor de *padd menor a 0.5* controlamos que no se genere un cromosoma excesivamente grande que resulta ineficiente a medida que pasan las generaciones. A continuación se muestra el mejor resultado obtenido con 2000 generaciones.

Figura 3: *La imagen superior izquierda muestra la imagen original, la superior derecha la imagen final reproducida (Uniform+SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:2,000), y la inferior muestra la gráfica de evolución del fitness.*



Por otro lado, se realizó una prueba con el objetivo de probar el desempeño del operador "Roulette" vs el de "Tournament". Para esto, se realizaron dos ejecuciones alternando los operadores y manteniendo constante el resto de la configuración.

Figura 4: *La imagen superior izquierda muestra la imagen original, la superior derecha la imagen final reproducida (Tournament + Uniform + SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:15,000), y la inferior muestra la gráfica de evolución del fitness.*

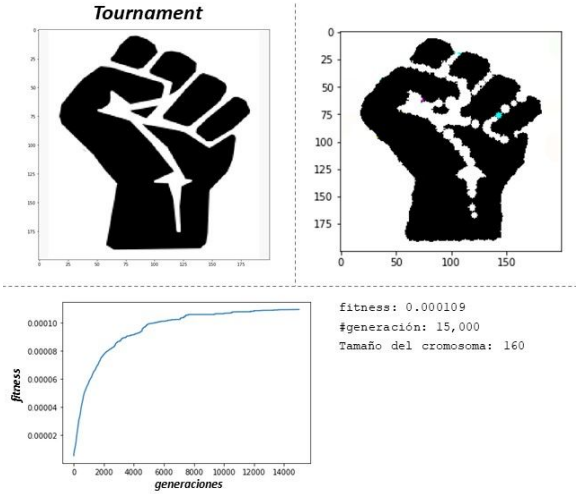
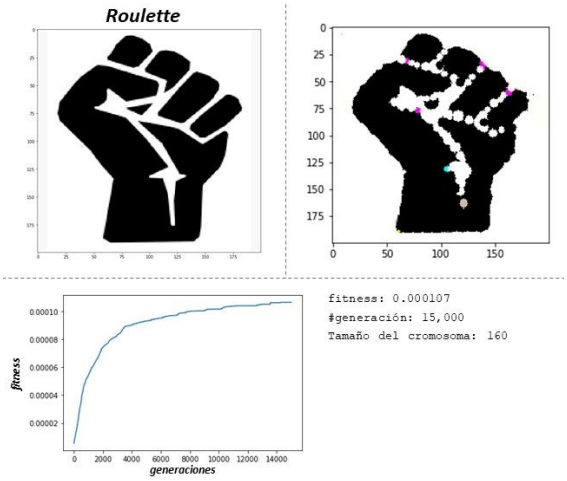


Figura 5: *La imagen superior izquierda muestra la imagen original, la superior derecha la imagen final reproducida (Roulette + Uniform + SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:15,000), y la inferior muestra la gráfica de evolución del fitness.*



Podemos observar de los resultados mostrados en la *Figura 4* y *Figura 5* que los fitness obtenidos por ambos operadores de selección son muy similares y el tamaño de sus mejores cromosomas es el mismo. Sin embargo, para evaluar más a detalle ambos operadores, se requiere también tomar en cuenta cómo cada uno se desempeña trabajando en conjunto con otros operadores de mutación y cruzamiento. Como en el alcance inicial del trabajo no se contempló realizar esta comparación, nos quedamos con el método de selección por torneo.

Finalmente, se realizaron pruebas ejecutando el AG con 15,000 generaciones para observar el desempeño del algoritmo trabajando con imágenes de diferentes características.

Figura 6: *La imagen superior izquierda muestra la imagen original (Mario Bros), la superior derecha la imagen final reproducida (Uniform+SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:15,000), y la inferior muestra la gráfica de evolución del fitness.*

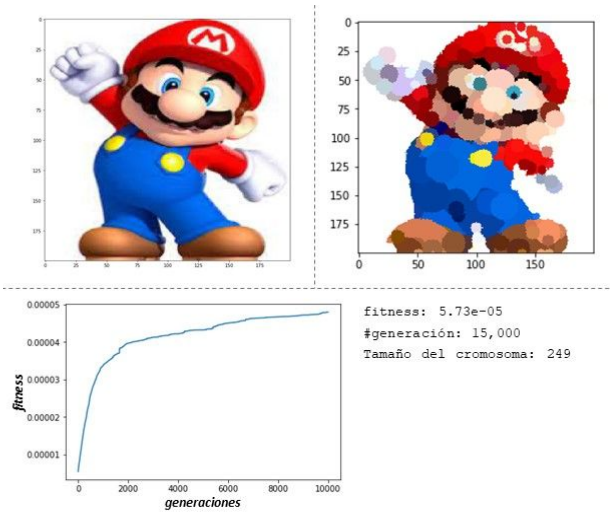


Figura 7: *La imagen superior izquierda muestra la imagen original (Machu Picchu), la superior derecha la imagen final reproducida (Uniform+SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:15,000), y la inferior muestra la gráfica de evolución del fitness.*

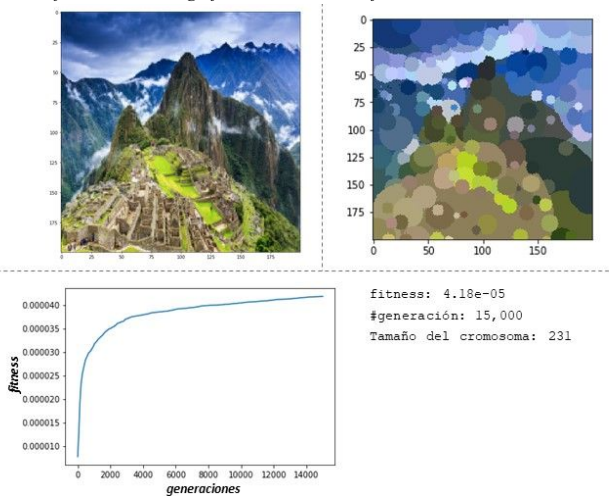
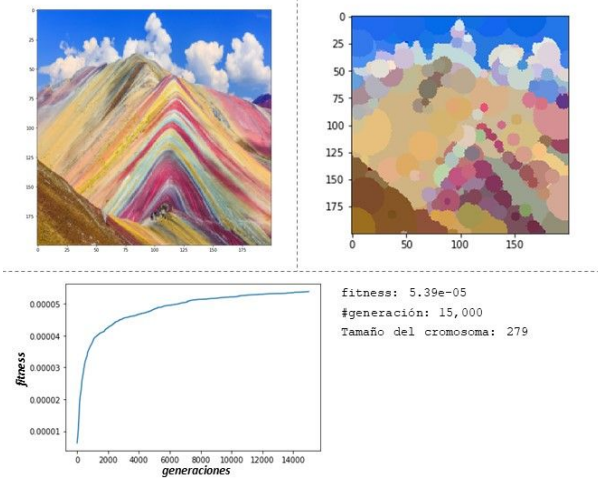


Figura 8: *La imagen superior izquierda muestra la imagen original (Montaña de Colores), la superior derecha la imagen final reproducida (Uniform+SingleGene, pmut:1.0, paddgene:0.2, imut:0.7, gen:15,000), y la inferior muestra la gráfica de evolución del fitness.*



4. Conclusión

En base a las pruebas realizadas y los resultados presentados previamente, podemos concluir los siguientes puntos.

i) Los valores de los fitness alcanzados no se acercan al valor óptimo que la *Fórmula 4* plantea (fitness=100) para las 15,000 generaciones probadas. Sin embargo, como se mencionó al inicio, el objetivo del AG con estilo puntillista no es obtener la misma imagen, sino representarla con círculos (al estilo puntillista), por lo que el fitness ha cumplido su propósito de guía y es por ello que podemos observar que las imágenes reproducidas (*Figura 6, Figura 7, y Figura 8*) tienden a asemejarse a las imágenes originales. Con una mayor cantidad de generaciones, es posible aumentar más el valor del fitness y por lo tanto lograr mayor detalle en las imágenes reproducidas.

ii) Debido a que la diferencia entre la imagen que define la población inicial y la imagen a replicar es muy grande, el AG se beneficia mucho de trabajar con la máxima probabilidad de mutación (*pmut=1.0*) para realizar un proceso de exploración más brusco. Sin embargo, se encontró mayor beneficio trabajando con mutación "Single-Gene" que con "All-Genes", probablemente porque el primer operador realiza cambios más leves que el segundo.

iii) La mejor combinación de operadores genéticos según las pruebas realizadas son, cruzamiento "Uniform" junto a mutación "SingleGene", con una probabilidad *pmut* máxima (1.0), una probabilidad de añadir genes menor a 0.5 (0.3-0.2) y una intensidad de mutación elevada entre (0.5 y 0.8). Según los resultados de las pruebas, aumentar sobre 0.5 la probabilidad de añadir un nuevo gen al cromosoma no es recomendable ya que el fitness no mejora e incluso puede empeorar, aumentando el tiempo de procesamiento. Esto debido a que el AG tendría que lidiar con una mayor cantidad de posibles combinaciones y modificaciones.

iii) Trabajando únicamente con operadores de cruzamiento o mutación no se obtuvieron buenos resultados. El primero debido a que los AGs necesitan de un componente de exploración, y este problema en particular lo demanda especialmente al inicio. El segundo, porque las soluciones encontradas en los mejores individuos no están siendo utilizadas para mejorarlas y en muchos casos se pierde el camino correcto. Representación de esto es la *Figura 1.b.* que presenta una curva incremental pero irregular.

iv) Podemos observar que las imágenes de referencia de la *Figura 3, Figura 4, Figura 5, Figura 6* son las que mejor se ven reproducidas por el AG, esto se debe a que no presentan muchos detalles, y la diversidad de colores es limitada. Sin embargo, las imágenes de referencia *Figura 7, Figura 8* al ser fotografías de mucho detalle, tienen una aproximación visual más lejana. Esto se debe además a que el AG tiene que modificar cada uno de los pixels con mayor cuidado y con reiterados intentos ya que trabaja únicamente con círculos. Por lo tanto, requiere una mayor cantidad de generaciones y aún así probablemente no se acerque mucho a la convergencia debido a la limitación de trabajar con círculos (característica que no bajo el enfoque del problema no es limitación, sino requerimiento).

5. Sugerencias de trabajos futuros

Tomando como referencia el AG implementado, se pueden considerar los siguientes pasos:

i) Si se desea mejorar la representación de las imágenes en menor tiempo, se puede trabajar en disminuir el espacio de estados, acotando los valores de color a escala de grises.

ii) Aplicar algoritmos de Inteligencia Colectiva para comparar valores de fitness, tiempos de ejecución y resultados visuales.

6. Link del repositorio del trabajo

Link de repositorio Github:

https://github.com/DavidFosca/Genetic_Algorithm_Painting.git

Donde se puede encontrar lo siguiente: a) Reporte de trabajo, b) Presentación, c) Notebook de Ejecución de Pruebas, d) Archivo Utils de funciones del AG, e) Tablas de Resultados, f) Videos de Resultados.

7. Referencias

[1]. Y. Zhao y D. Xu, "Monet-style images generation using recurrent neural networks", E-Learning and Games, Springer International Publishing, 2016, pp. 205–211.

[2]. A. Kasao y M. Nakajima, "Synergistic image creator? a picture generation system with consideration of the painting process" vol. 30, pp. 13–21, 09 1999.

[3]. A. Hansen y C. Lewis. "Applying Genetic Algorithms to Generating Paintings", Int'l Conf. IP, Comp. Vision, and Pattern Recognition, 2018, pp. 92-98.

[4]. R. Johansson, "Genetic programming: Evolution of mona lisa," 2008. Disponible en: rogerjohansson.blog/2008/12/07/genetic-programming-evolution-of-mona-lisa/

[5]. D. Estupiñan, D. Machado, A. Pazmiño, "Manual Técnico de Puntillismo para Estudiantes de Primero y Segundo Nivel de Diseño Gráfico". 2019. Disponible en: http://repositorio.uisrael.edu.ec/handle/47000/2205