

# **Simplificación de mallas 3D utilizando algoritmos genéticos**

**Maria Isabel Arango Palacio, María José Bernal Vélez**

**Isabella Montoya Henao**

Universidad EAFIT

Medellín, Colombia

## **Resumen**

Muchas de las aplicaciones de los modelos 3D en diferentes áreas requieren modelos de superficies poligonales con un alto nivel de detalle, esto implica una complejidad alta en el procesamiento del modelo y a su vez una capacidad de computo y almacenamiento grande. Es por esto que, la simplificación de mallas poligonales tiene como objetivo reducir el número de polígonos del modelo, conservando su forma. En el presente trabajo, se hace uso del algoritmo genético NSGA-II para desarrollar el problema de simplificación de mallas poligonales 3D, para esto se plantean como objetivos, maximizar la precisión y la simplicidad del modelo. Así, se utilizan diferentes modelos 3D asociados a rostros para implementar y después comparar los resultados y el rendimiento del algoritmo NSGA-II al utilizar dos métricas diferentes para medir el error. Los resultados experimentales demuestran que el algoritmo implementado es una muy buena herramienta para solucionar el problema, permite obtener diferentes configuraciones de mallas que cumplen con los objetivos.

## **1. Introducción**

Los modelos de superficies poligonales son comúnmente usados en la visualización y simulación 3D en diferentes áreas, como el diseño asistido por computador, visión por computador, medicina, topografía, automatización, entre muchas otras. Generalmente, estos modelos son usados para ilustrar objetos con un gran nivel de detalle, por lo que dichas superficies poligonales están compuestas de millones de polígonos, lo que hace que su procesamiento sea muy complejo y se requiera de alta capacidad de cómputo y almacenamiento [1]. Es por esto que surge la necesidad de desarrollar diferentes técnicas que permitan simplificar la malla, de manera que se reduzca el número de polígonos que esta contiene, manteniendo su forma lo mejor que se pueda.

En el presente trabajo, se hace uso de técnicas de optimización multiobjetivo, más específicamente de algoritmos evolutivos para simplificar modelos de superficies poligonales asociados a imágenes de rostros. De esta manera, se

obtienen diversas alternativas que reducen el modelo 3D, las cuales satisfacen los dos objetivos principales que son maximizar la precisión del modelo junto con su simplicidad. Así pues, el trabajo a desarrollar está compuesto en una primera parte por el marco teórico, en donde se introducen conceptos teóricos fundamentales para el desarrollo del trabajo. Posteriormente, se presenta la metodología, allí se describe como se pretende resolver el problema. Finalmente, se presentan los resultados obtenidos al utilizar un algoritmo genético con diferentes métricas para simplificar los modelos de malla poligonales.

## 2. Objetivos

### 2.1. Objetivo general

Optimizar una malla 3D asociada a una imagen de rostro, usando algoritmos genéticos con el objetivo de reducir la cantidad de polígonos que conforman la discretización del objeto 3D y conservar su topología.

### 2.2. Objetivos específicos

- Discretizar una imagen, utilizando la triangulación de Delaunay para facilitar la optimización de la malla 3D.
- Implementar el algoritmo genético NSGA-2 para resolver el problema de optimización multiobjetivo planteado, utilizando diferentes métricas para calcular el error.
- Analizar los resultados obtenidos con el algoritmo NSGA-II al utilizar diferentes métricas y modificaciones en los parámetros del algoritmo.

## 3. Marco teórico

### 3.1. Triangulación

La triangulación es la división de una superficie o polígono plano en un conjunto de triángulos, la cual generalmente tiene la restricción de que todos los lados de los triángulos son compartidos por completo por dos triángulos adyacentes [2]. Es decir, una triangulación  $T$  es una partición de un dominio  $\Omega$  de puntos  $P$ , en una colección de triángulos con los siguientes requisitos:

- No existe ningún triángulo  $\Delta_{ijk}$  en la triangulación  $T$  que sea degenerado. Esto es, los puntos  $p_i, p_j, p_k$  que conforman el triángulo no son colineales.
- Los interiores de dos triángulos cualquiera no se intersectan.
- Los límites de dos triángulos cualquiera de la triangulación solo pueden cruzarse en un vértice o borde común.

- La unión de todos los triángulos formados en la triangulación es igual al dominio  $\Omega$  sobre el cual se define  $T$ .
- La triangulación no puede dejar ningún punto  $p_i$  sin triangular.
- Para todo vértice  $v_i$  en la frontera del dominio  $\Omega$  existen exactamente dos aristas de contorno que contienen a  $v_i$  como un vértice común [3].

Así pues, dado un conjunto de puntos, existen muchas triangulaciones posibles, las cuales se distinguen por sus características y su construcción.

### 3.1.1. Triangulación de Delaunay

La triangulación de Delaunay es una de las triangulaciones más conocidas e utilizadas, debido a los grandes beneficios teóricos que posee y la facilidad de su construcción [3].

Dado un conjunto de puntos  $P$  en un plano, la triangulación de Delaunay es una triangulación  $T_d$  con la condición de que cada circuncentro de un triángulo sea un círculo vacío (no contenga ningún otro punto del conjunto) [4]. De esta manera, la triangulación de Delaunay maximiza el ángulo mínimo de todos los triángulos (MaxMin angle), y simultáneamente, minimiza el ángulo máximo (MinMax angle). Esto último permite que los triángulos que conforman a  $T_d$  sean lo más similares a triángulos equiláteros, evitando triángulos alargados [1]. A continuación se presenta de manera gráfica la triangulación de Delaunay:

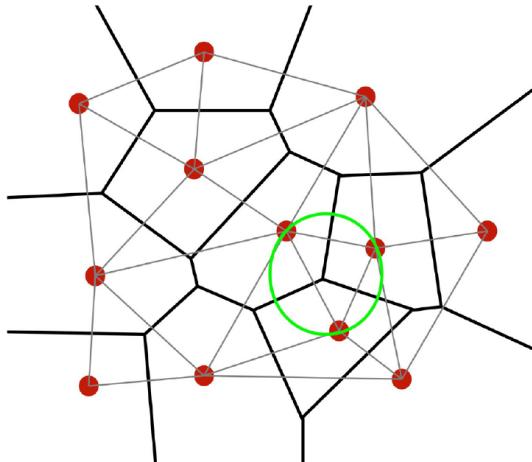


Figura 1: Triangulación de Delaunay [5]

Algunas de sus propiedades son:

- Es única siempre y cuando el conjunto  $P$  sea no degenerado (no hay 3 puntos en la misma línea ni 4 en una circunferencia).
- Para cada triángulo  $\triangle ijk$ , el círculo que une sus vértices  $v_i, v_j, v_k$  no contiene ningún otro punto de  $P$  en su interior.

- Al maximizar el ángulo más pequeño, no posee muchos triángulos largos y angostos, y tiene en su mayoría triángulos parecidos a equiláteros.
- La suma de los radios de todos los círculos contenidos en la triangulación es mínima [3].

Ahora bien, la importancia de evitar triángulos largos y angostos es que estos requieren realizar una interpolación entre puntos muy alejados, lo que puede resultar en errores considerables en los resultados [3].

### 3.2. Algoritmos genéticos

Alrededor del año 1970, John Henry Holland junto con sus colegas y sus estudiantes en la Universidad de Michigan desarrollaron una de las líneas mas prometedoras de la inteligencia artificial, los algoritmos genéticos, los cuales son algoritmos de búsqueda y optimización basados en los mecanismos de la selección natural y la genética. Estos algoritmos eligen a los individuos más aptos de cada generación para crear a los individuos que conformarán la población siguiente, haciendo cambios aleatorios en algunos de ellos.

No obstante, no es un algoritmo meramente aleatorio, pues hace uso eficazmente de la información histórica para determinar nuevos puntos con un rendimiento mejorado [6]. Adicionalmente, estos algoritmos han demostrado una gran efectividad a la hora de aproximar la frontera de Pareto de los problemas multiobjetivo.

Los algoritmos genéticos pueden tener variaciones según la forma en la que se realice la selección, cruzamiento y mutación de los individuos que van a formar la generación de descendientes. De manera general, estos siguen los siguientes pasos:

1. **Inicialización:** El algoritmo comienza creando una población inicial aleatoria, que está constituida por un conjunto de individuos los cuales representan las posibles soluciones del problema.
2. **Evaluación:** Se cuantifica qué tan bueno es cada individuo como solución al problema, la cual se le llama fitness. Dependiendo del tipo de de un problema de maximización o minimización, la relación del fitness con la función objetivo  $f$  puede ser:
  - *Maximización:* el individuo tiene mayor fitness mientras mayor sea el valor de la función objetivo.
  - *Minimización:* el individuo tiene mayor fitness mientras menor sea el valor de la función objetivo.
- a) **Selección:** Después de saber la aptitud de cada individuo se procede a elegir los individuos que serán cruzados en la siguiente generación, los cuales suelen ser los más aptos. Generalmente, los más probables de ser seleccionados son las que cuentan con un mayor fitness.
- b) **Cruce:** Busca combinar dos individuos (progenitores) para generar uno o más descendientes donde se combinan las características de ambos. Para el cruzamiento se pueden seguir diversas estrategias, donde las mas empleadas son cruzamiento a partir de un solo punto, cruzamiento a partir de múltiples puntos y cruzamiento uniforme.

- c) **Mutación:** Modifica aleatoriamente parte de los individuos de la población, y permite alcanzar soluciones que no estaban cubiertas por los individuos de la población actual [7]
3. **Condición de término:** El algoritmo se ejecuta hasta que se alcance un número máximo de iteraciones (generaciones) o cuando no haya cambios en la población. Mientras no se cumpla la condición de término se continua realizando el proceso anterior.

### 3.3. NSGA-II

El NSGA-II es uno de los algoritmos genéticos más importantes y reconocidos en el ámbito de la optimización multiobjetivo. Fue propuesto en el año 2002 por un grupo de investigación liderado por el reconocido informático hindú, Kalyanmoy Deb, como una versión mejorada del algoritmo original NSGA [8]. Esta versión fue desarrollada como respuesta a algunas de las deficiencias de los primeros algoritmos evolutivos como la alta complejidad computacional, la falta de elitismo, la necesidad de un parámetro compartido para mantener la diversidad en la solución, entre otras. Es por esto que, el NSGA-II se caracteriza por las siguientes tres cosas:

1. La clasificación rápida no dominada, la cual permite reducir significativamente la complejidad.
2. El elitismo, el cual permite que las mejores soluciones sean conservadas a través de las diferentes generaciones y además aumenta la velocidad de convergencia del algoritmo [9].
3. El operador simple de comparación entre los individuos de la población.

La principal idea del algoritmo es encontrar un conjunto de soluciones en la población, las cuales evolucionen hacia la solución óptima para resolver un problema de optimización multiobjetivo. El NSGA-II está diseñado para encontrar un conjunto óptimo de soluciones no dominadas, un conjunto de Pareto [10]. De esta manera, el algoritmo se describe a continuación:

- **Paso 1: Inicialización de la población**

En primera instancia, se crea una población aleatoria  $P_0$  de individuos, de tamaño N. Luego, se aplica un cruce y mutación a esta población y de ahí se genera la población descendiente  $Q_0$ , de tamaño N. Así, la población inicial total será  $R_t = P_0 \cup Q_0$ , de tamaño 2N.

- **Paso 2: Ordenamiento rápido no dominado**

Cada individuo (solución) de la población  $R_t$  es comparado con todos los otros para determinar si es dominado por alguno, o si por el contrario es el que domina a los demás. Según el resultado, los individuos son clasificados, aquellos con más dominancia en el sentido de pareto son los que mejor rango de clasificación tendrán. Así, de acuerdo con esta clasificación son construidos cada uno de los frentes de no dominancia  $F_1, F_2, \dots, F_l$  del algoritmo, los frentes de pareto. De esta manera, en el primer frente  $F_1$  se encuentran los individuos que no son

dominados por ningún otro y su rango es 1, esto quiere decir que, en  $F_1$  se encuentran las mejores soluciones y son aquellas que deben ser priorizadas por encima de cualquier otra solución. Y así sucesivamente con cada frente. Este ordenamiento se realiza en las siguientes dos etapas:

- **Etapa 1:** para cada individuo  $p$  de la población, se asignan las variables  $S_p$  y  $n_p$ , que corresponden al conjunto de elementos que son dominados por  $p$ , y la cantidad de individuos que dominan a  $p$ , respectivamente. Luego, los elementos  $p_i$  cuyo atributo  $n_i$  sea 0, lo que indica que no es dominado por ningún otro elemento, son agregados al primer frente  $F_1$  y se les asigna un fitness (posición) de 1 [11].
- **Etapa 2:** ahora, se realiza un ciclo para establecer los elementos los demás frentes  $F_2, F_3, \dots, F_{i+1}$ , el cual es ejecutado mientras que el frente anterior  $F_i$  no esté vacío. De este modo, se comienza por llenar el frente  $F_{i+1}$  recorriendo los elementos  $p$  que están en el frente  $F_i$  y los individuos  $q$  que están en el atributo  $S_p$  de cada elemento de  $f_i$ . Luego, cada valor  $n_q$  es reducido en una unidad, indicando que, sin tener en cuenta el frente  $F_i$ , el elemento  $q$  ya no es dominado por los elementos de ese frente. De la misma manera, al terminar esta revisión, los elementos  $q$  cuyo atributo  $n_q$  sea 0, son agregados al frente  $F_{i+1}$  y se les asigna un fitness de  $i + 1$ . Este mismo proceso se repite hasta que todos los individuos sean agregados a algún frente [11].

#### ■ **Paso 3: Selección**

Es importante reconocer que en general, el conjunto de todas las soluciones desde  $F_1$  hasta  $F_l$  será mayor que el tamaño de la población  $N$ , ya que estos se obtienen a partir de la combinación de las poblaciones  $P$  y  $Q$ . Es por esto que, se establecen ciertos criterios para obtener la población  $P_{t+1}$  a partir de cada uno de los frentes. En primer lugar, se evalúa si el tamaño del frente uno es menor que  $N$  (tamaño de la población), si esto pasa, se escogen todos los individuos de dicho frente para ser parte de la población. Los miembros restantes se escogen de los siguientes frentes no dominados según su rango, esto sucede hasta que ningún otro frente pueda ser acomodado en su totalidad en la población. Cuando esto sucede, y aún faltan individuos para tener el tamaño deseado de la población ( $N$ ), se usa una selección de torneo binaria con el operador de comparación multitudes ( $\prec_n$ ), el cual se describe a continuación [12].

- **Operador de comparación de multitudes:** Este operador es usado para el proceso de selección de individuos del algoritmo. Asume que cada individuo  $i$  de la población tiene dos atributos:

1. Rango de no dominancia ( $i_{rango}$ )
2. Distancia de apilamiento ( $i_{dist}$ )

De acuerdo con esto, se define la relación parcial ( $\prec_n$ ) como:  $i \prec_n j$  si:

1.  $i_{rango} < j_{rango}$  o
2.  $i_{rango} = j_{rango}$  y  $i_{dist} > j_{dist}$

Entonces, de acuerdo a lo descrito anteriormente, para cada individuo del frente que no puedo ser seleccionado en su totalidad, se calcula la distancia de apilamiento, la cual se define como la suma de las distancias del punto  $i$  con todos los demás. Esta garantiza la diversificación en las soluciones. De forma geométrica, esta distancia denota la mitad del perímetro que está encerrado por las soluciones vecinas más cercanas del mismo frente [13]. Para calcularla, se inicializa la distancia  $d_j = 0$ , con  $j = 1, \dots, m_i$ , para los  $m_i$  individuos de cada frente  $F_i$ . Luego, para cada función objetivo  $f_k$ , ordena los individuos del frente en orden ascendente, y asigna distancias infinitas al individuo con menor y mayor valor de  $f_k$  [14]. Así, para los demás individuos, calcula la distancia como:

$$d_j = d_j + \frac{|f_k(j-1) - f_k(j+1)|}{f_k^{\max} - f_k^{\min}}$$

donde  $f_k(j-1)$  y  $f_k(j+1)$  son los valores que toma la función para los elementos inmediatamente anterior y posterior del elemento cuya distancia está siendo calculada, respectivamente, y  $f_k^{\max}$  y  $f_k^{\min}$  son los valores máximo y mínimo que toma la función en ese frente, respectivamente. Esto mismo se repite para cada una de las funciones objetivo, y se va acumulando la distancia de cada individuo [14].

Posteriormente, se aplica el operador para establecer la relación parcial  $\prec_n$  entre los individuos y poder así seleccionar los mejores para completar la población  $P_{t+1}$ .

#### ■ Paso 4: Recombinación y selección

La nueva población  $P_{t+1}$  es ahora usada para la selección (selección torneo binaria), el cruce y la mutación para generar la población descendiente  $Q_{t+1}$ . Luego, se vuelven a combinar las poblaciones  $R_t = P_0 \cup Q_0$  y se vuelve al paso 2 hasta que se hayan conseguido el criterio de parada, el cual generalmente esta relacionado con las generaciones deseadas.

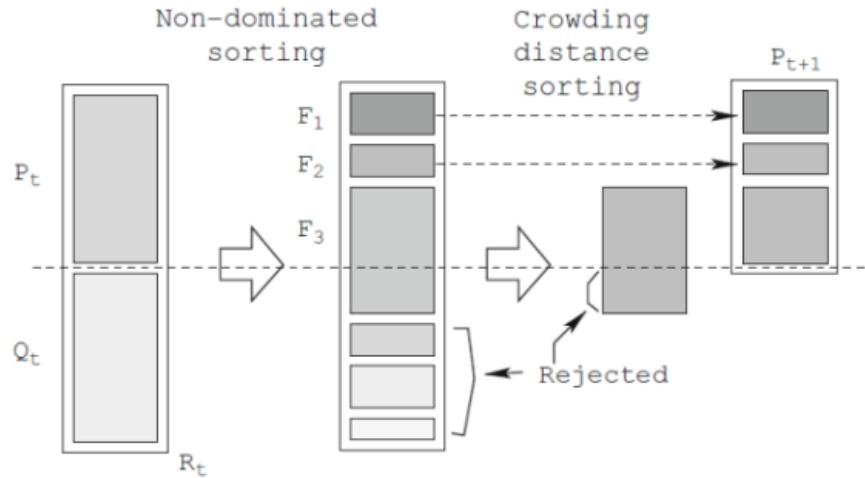


Figura 2: Esquema del algoritmo NSGA-II [11]

## 4. Metodología

### 4.1. Planteamiento del problema

En el presente trabajo se han considerado dos objetivos a maximizar, los cuales son la precisión y la simplicidad. No obstante, para lograr una mejor formulación, se plantean los siguientes:

$$\min F_1(m) = T(m)$$

$$\min F_2(m) = E(m)$$

donde  $T(m)$  se refiere a la cantidad de triángulos que conforman la malla poligonal, y  $E(m)$  es el error de la simplificación.

Ahora, para calcular  $F_2(m)$  se hace uso de dos métricas diferentes, con el fin de establecer cual permite una mayor precisión en la simplificación de la malla. Estas son:

- **Error en el área superficial:** se define como

$$F_2^1(m) = |A(m) - A(\hat{m})|$$

donde  $A(m)$  es el área superficial de la malla original y  $A(\hat{m})$  es la de la malla simplificada. En general, el área está dada por:

$$\begin{aligned} A(m) &= \sum_{t \in m} a(t) \\ &= \sum_{t \in m} \frac{1}{2} |x_t| |y_t| \sin \theta_t \end{aligned}$$

donde  $x_t$  y  $y_t$  son dos lados del triángulo  $t$  y  $\theta_t$  es el ángulo entre ellos.

- **Error en los valores de las alturas:** está definido por:

$$F_2^2(m) = \sum_{v \in m} |z_v - \hat{z}_v|$$

donde  $z_v$  es el valor de la altura del vértice  $v$  en la malla original y  $\hat{z}_v$  el valor de la altura al realizar la interpolación de  $v$  en la nueva malla. De esta manera, el error se define como la suma de las distancias sobre el total de vértices en la malla inicial.

Cabe resaltar que para realizar la interpolación, para cada vértice  $v$  de la malla original se determina a qué

triángulo de la malla simplificada  $\hat{m}$  pertenece. Luego, se encuentra la ecuación del plano formado por los tres vértices del triángulo encontrado, para así hallar la tercer componente  $\hat{z}_v$ .

## 4.2. Algoritmo NSGA-II

Para lograr la solución del problema multiobjetivo planteado anteriormente se hace uso del algoritmo NSGA-II, ya que este permite obtener diferentes configuraciones de mallas poligonales que alcancen los objetivos planteados. Para este problema, los individuos del algoritmo NSGA-II se definen como subconjuntos de la malla original, los cuales tienen como atributos los vértices y la triangulación de dichos vértices (caras). A continuación se presenta el desarrollo del algoritmo:

- **Inicialización de la población:** se define una población inicial de 50 individuos, los cuales tienen entre 3000 y 7000 vértices de la malla original que son elegidos aleatoriamente. Adicionalmente, para conservar la topología de la malla original, se conservan 14 vértices del borde en todos los individuos. Además, sus caras son calculadas utilizando la triangulación de Delaunay.
- **Cruzamiento:** se seleccionan dos padres aleatoriamente para generar un nuevo individuo para obtener la población descendiente. En primer lugar, se determina el tamaño del individuo. Luego, mediante un parámetro de cruzamiento se elige la cantidad de vértices que este va a tener de cada parente.
- **Mutación:** de acuerdo a la probabilidad de mutación, la cual se recomienda que sea menor al parámetro de cruzamiento, se determina si un individuo será mutado o no. En caso de serlo, el 50 % de las veces se le eliminará el 5 % de sus vértices de manera aleatoria; de lo contrario, se aumenta su tamaño en un 5 % con algunos de los vértices de la malla original que no contiene.

De esta manera, se siguen los pasos descritos en el algoritmo NSGA-II para obtener los resultados utilizando la inicialización, el cruzamiento y la mutación anteriores.

## 5. Resultados

Para obtener los resultados del algoritmo NSGA-II utilizado, se hace uso de la siguiente malla poligonal, la cual ilustra la cara de un hombre [15]. Originalmente, el modelo 3D contenía la cabeza completa junto con el cuello, pero esta fue recortada usando el software *Blender* para tener únicamente la parte frontal de la cara. Esta malla contiene 10049 vértices, 20058 caras, tiene un peso de 1.9MB, y su área original es de 10.2012 unidades cuadradas. A continuación se puede observar la malla original:



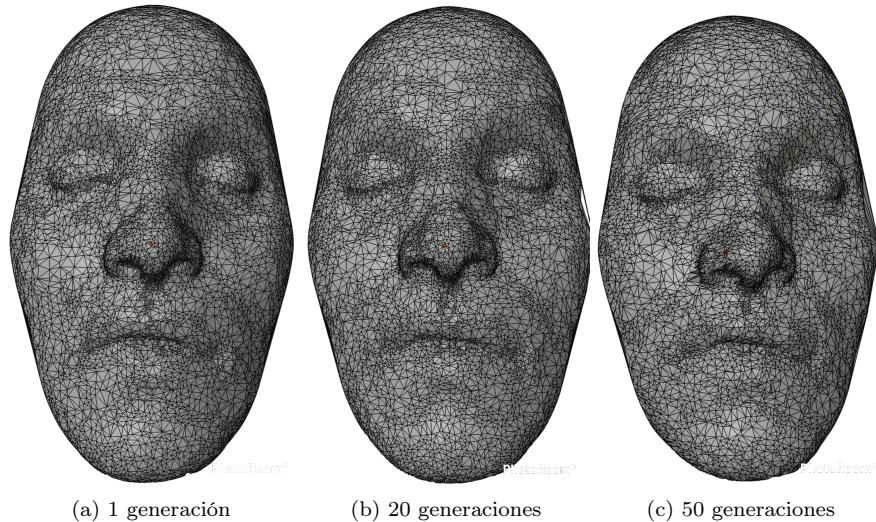
Figura 3: Malla de la cara original

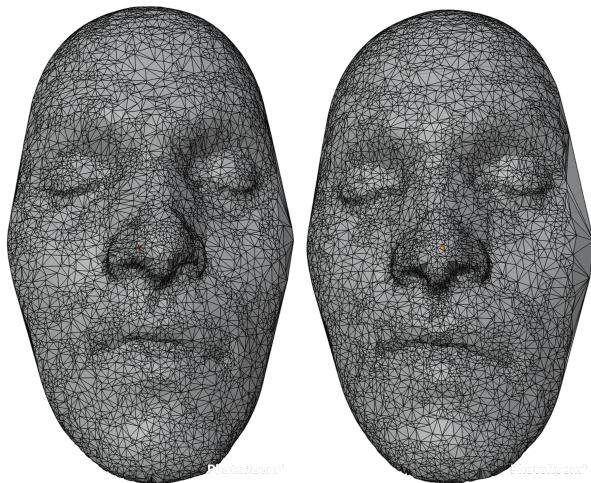
### 5.1. Modificación en las generaciones

Ahora, se analiza qué cambios tiene la solución del algoritmo implementado al modificar la cantidad de generaciones en 1, 20, 50, 100 y 200. Para esto, se resuelven las dos versiones propuestas para el problema multiobjetivo tomando el parámetro de cruzamiento como 0.5 y la probabilidad de mutación 0.1.

## Error en el área superficial

Al tener como funciones objetivo la cantidad de triángulos de la malla y el error en el área superficial, se obtienen los siguientes resultados en la simplificación de la malla:

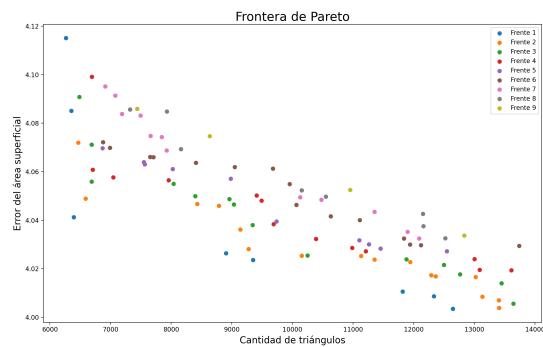




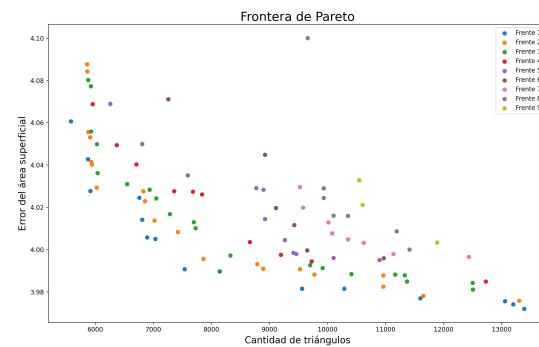
(d) 100 generaciones

(e) 200 generaciones

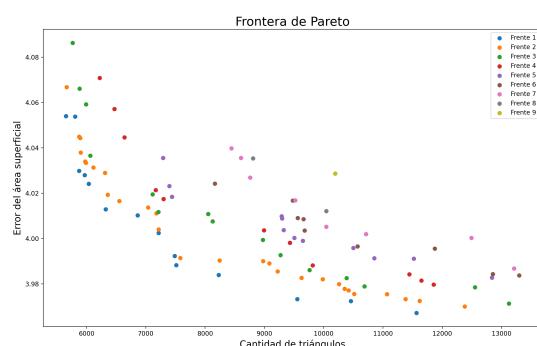
Adicionalmente, los frentes de Pareto obtenidos para cada una de estas variaciones de la cantidad de generaciones se ilustran a continuación:



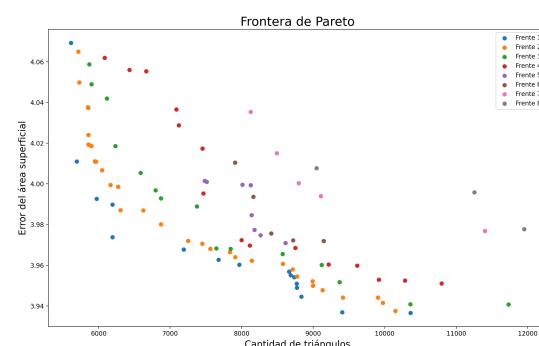
(a) 1 generación



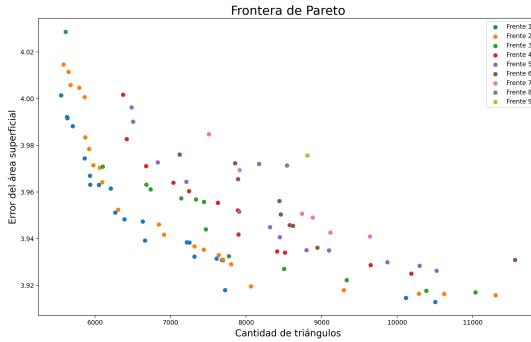
(b) 20 generación



(c) 50 generación



(d) 100 generación



(e) 200 generación

De los resultados anteriores se puede ver que, a medida que aumenta la cantidad de generaciones, se distinguen de manera más clara cada uno de los frentes de Pareto. Adicionalmente, se puede evidenciar que el algoritmo conserva las mejores soluciones, ya que hay puntos del frente 1 que se conservan en todas las generaciones analizadas, dejando ver el elitismo, una de las características principales del algoritmo NSGA-II. Finalmente, con el aumento de la cantidad de generaciones, la cantidad de individuos que se clasifican en el frente 1 va incrementando, lo que indica que cada vez se obtienen soluciones mejores y más diversas.

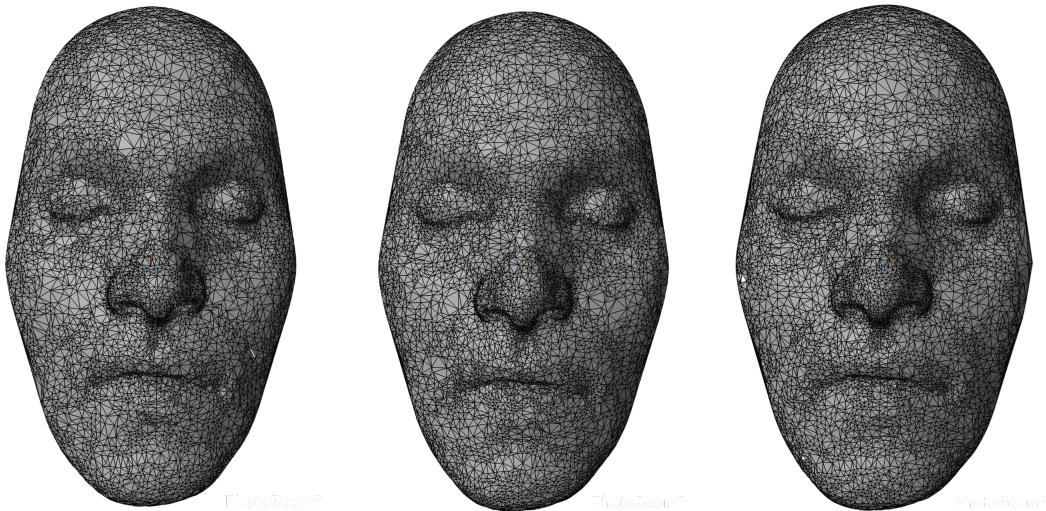
En la siguiente tabla se muestran los resultados obtenidos para el individuo del primer frente con la mejor distancia de apilamiento.

| Generación | No. vértices | No. triángulos | Error  | Tiempo de ejecución | Peso en memoria (KB) |
|------------|--------------|----------------|--------|---------------------|----------------------|
| 1          | 6460         | 12645          | 4.0034 | 00:18.71            | 403                  |
| 20         | 6830         | 13385          | 3.9719 | 01:37.33            | 426                  |
| 50         | 5899         | 11566          | 3.9672 | 03:36.77            | 367                  |
| 100        | 5293         | 10361          | 3.9365 | 06:34.67            | 329                  |
| 200        | 5367         | 10501          | 3.9129 | 12:53.63            | 333                  |

De la tabla anterior, se puede ver que el tiempo de ejecución aumenta notablemente a medida que se incrementa la cantidad de generaciones. No obstante, el peso en memoria de la malla comprimida disminuye, lo cual se debe a la disminución en la cantidad de triángulos. Ahora bien, es posible notar que al hacer 200 generaciones no se obtienen mejoras significativas con respecto a realizar 100 generaciones, por lo que no vale la pena tener esta cantidad de generaciones en cuenta, ya que el tiempo de ejecución es aproximadamente el doble que la anterior.

### Error en los valores de las alturas

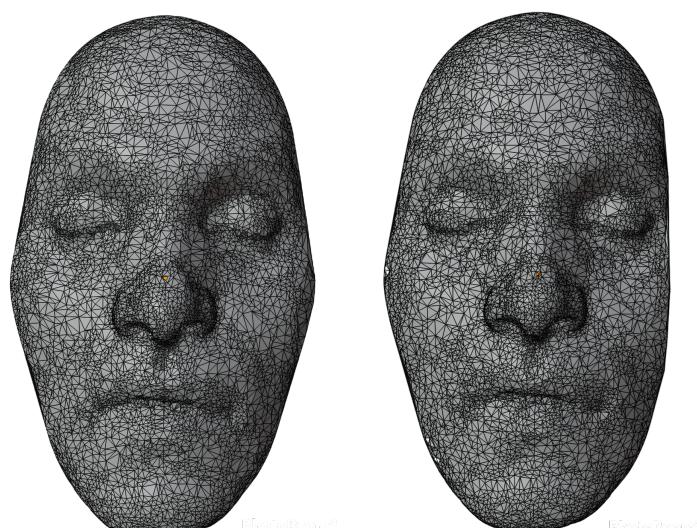
Por otra parte, se resuelve el problema multiobjetivo modificando el cálculo del error, el cual se calcula ahora como la suma de las diferencias absolutas entre la altura real y la interpolada. Con esto se obtienen los resultados que se muestran a continuación:



(a) 1 generación

(b) 20 generaciones

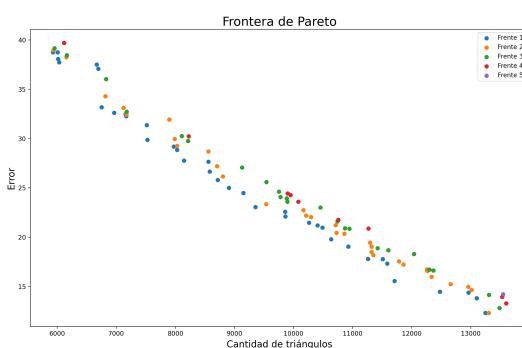
(c) 50 generaciones



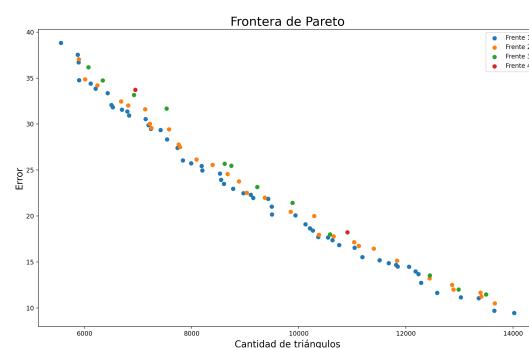
(d) 100 generaciones

(e) 200 generaciones

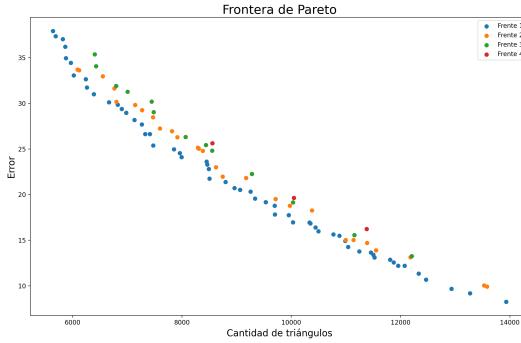
Las siguientes gráficas muestran los frentes de Pareto para cada valor en la cantidad de generaciones.



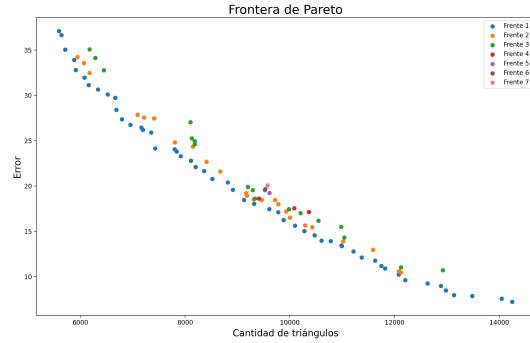
(a) 1 generación



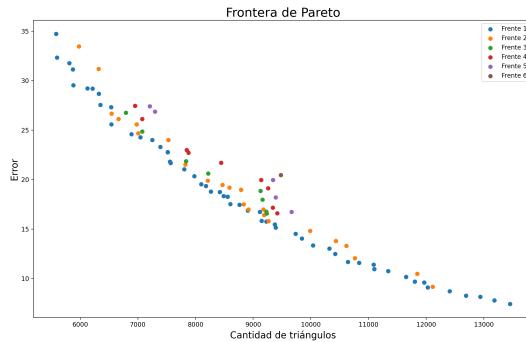
(b) 20 generación



(c) 50 generación



(d) 100 generación



(e) 200 generación

De las gráficas anteriores, se puede ver que a diferencia del cálculo del error anterior, desde la primera generación se forman menos frentes, lo cual indica que hay más individuos que no son dominados por los demás. Además, a medida que se incrementa la cantidad de generaciones, aumenta la cantidad de individuos en el primer frente, ya que se conservan los mejores con el paso de las generaciones, garantizando el elitismo y la diversidad en las soluciones.

Así, la siguiente tabla muestra los resultados para la malla con la mayor distancia de apilamiento en el frente 1:

| Generación | No. vértices | No. triángulos | Error   | Tiempo de ejecución | Peso en memoria (KB) |
|------------|--------------|----------------|---------|---------------------|----------------------|
| 1          | 6754         | 13246          | 12.3267 | 00:33.23            | 421                  |
| 20         | 7147         | 14019          | 9.4505  | 03:20.89            | 446                  |
| 50         | 7095         | 13930          | 8.2445  | 07:46.30            | 444                  |
| 100        | 7259         | 14243          | 7.2244  | 14:54.27            | 454                  |
| 200        | 6853         | 13450          | 7.4225  | 29:12.61            | 428                  |

De la tabla anterior, se puede ver que a medida que aumenta la cantidad de generaciones, el peso en memoria disminuye, lo cual se debe a que se reduce la cantidad de vértices y con esto la cantidad de triángulos. Sin embargo, con el aumento de generaciones, el tiempo de ejecución incrementa notablemente, y puede verse que el tiempo de ejecución pasa a ser aproximadamente el doble que la función anterior.

Por otra parte, es posible notar que al hacer 200 generaciones se obtienen mejores resultados que en las otras generaciones. No obstante, es bastante costoso computacionalmente, y las mejoras no tienen un alto impacto en los

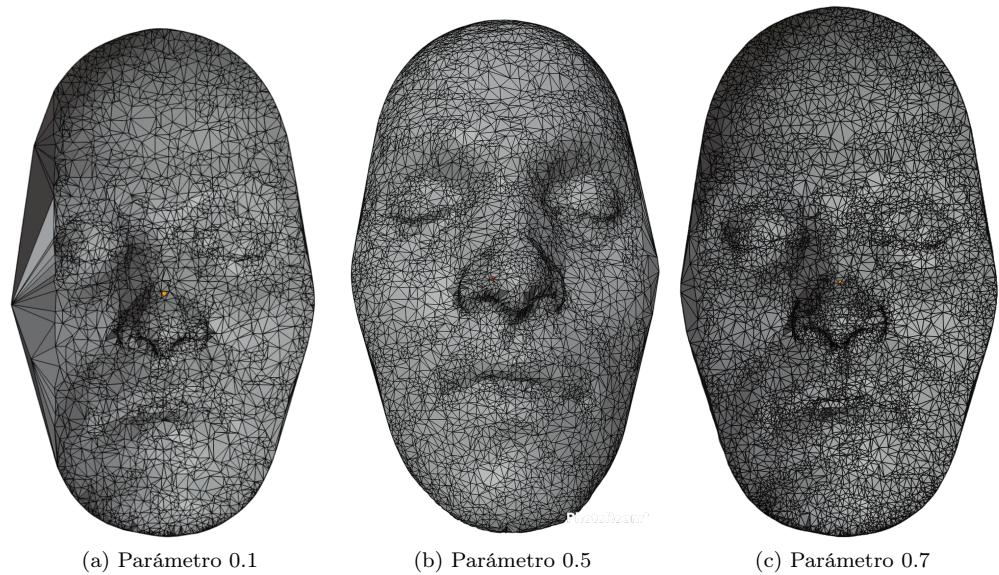
resultados, por lo que es preferible realizar 100 generaciones.

## 5.2. Modificación en el parámetro de cruzamiento

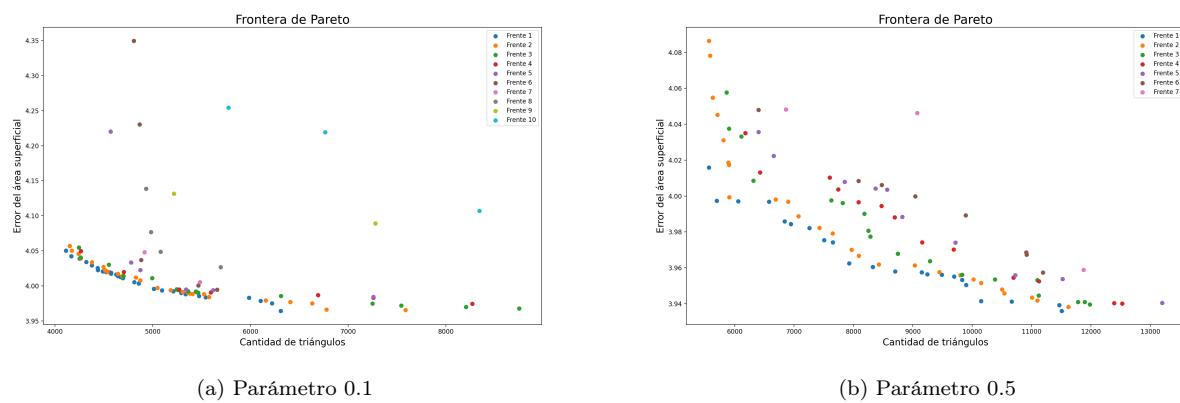
Para analizar el impacto que tiene el parámetro de cruzamiento en los resultados del algoritmo, se toman los valores de 0.1, 0.5 y 0.7 y se analiza lo obtenido con cada métrica de error. Para esto, se tomaron 100 generaciones de 50 individuos con una probabilidad de mutación fija de 0.1.

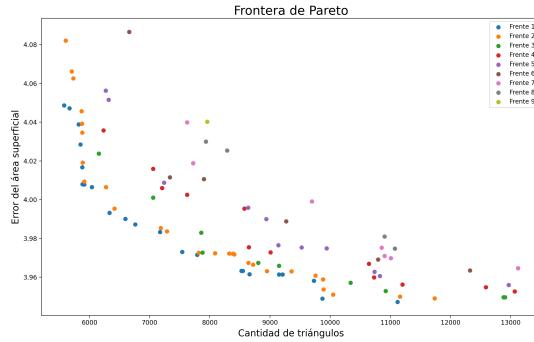
### Error en el área superficial

Teniendo como funciones objetivo minimizar la cantidad de triángulos de la malla simplificada y el error medido a través de la diferencia absoluta del área superficial, se obtienen los siguientes resultados para los diferentes valores del parámetro de cruzamiento:



Las fronteras de Pareto para estos valores del parámetro de cruzamiento son:





(c) Parámetro 0.9

Es posible notar que cuando el parámetro de cruzamiento es 0.1, las diferencias en los valores de las funciones objetivo del primer frente con los demás son muy pequeñas. Además, los valores del primer frente no toman un rango tan amplio de valores, como con las demás variaciones del parámetro de cruzamiento. Por otro lado, a medida que aumenta el parámetro de cruzamiento los frentes y las soluciones se vuelven más dispersos, logrando diferenciar más claramente cada uno de ellos.

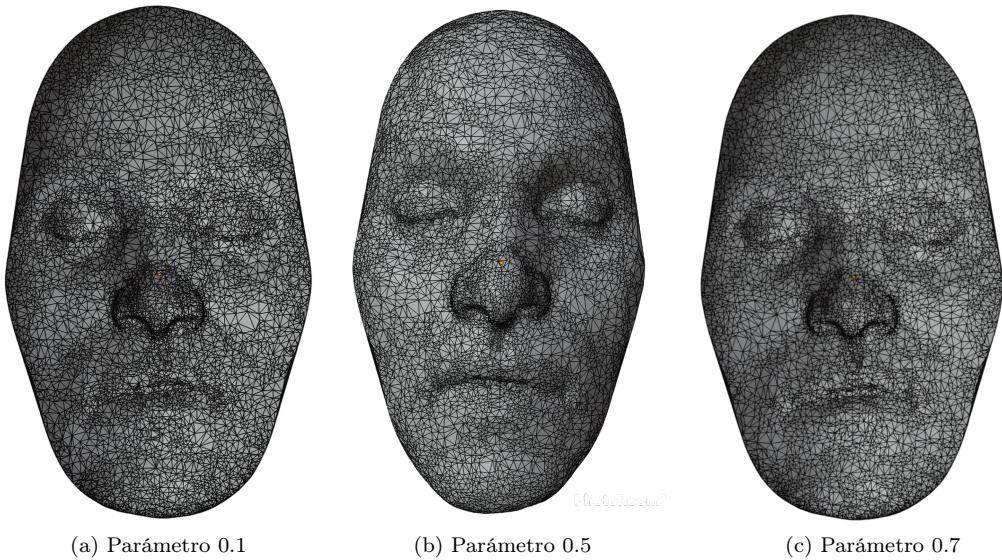
A continuación, se presentan los resultados para la malla con la mayor distancia de apilamiento en el frente 1:

| Parámetro de cruzamiento | No. vértices | No. triángulos | Error  | Tiempo de ejecución | Peso en memoria (KB) |
|--------------------------|--------------|----------------|--------|---------------------|----------------------|
| 0.1                      | 3220         | 6310           | 3.9641 | 04:20.48            | 198                  |
| 0.5                      | 5870         | 11506          | 3.9359 | 06:06.84            | 366                  |
| 0.7                      | 5668         | 11120          | 3.9472 | 06:21.37            | 353                  |

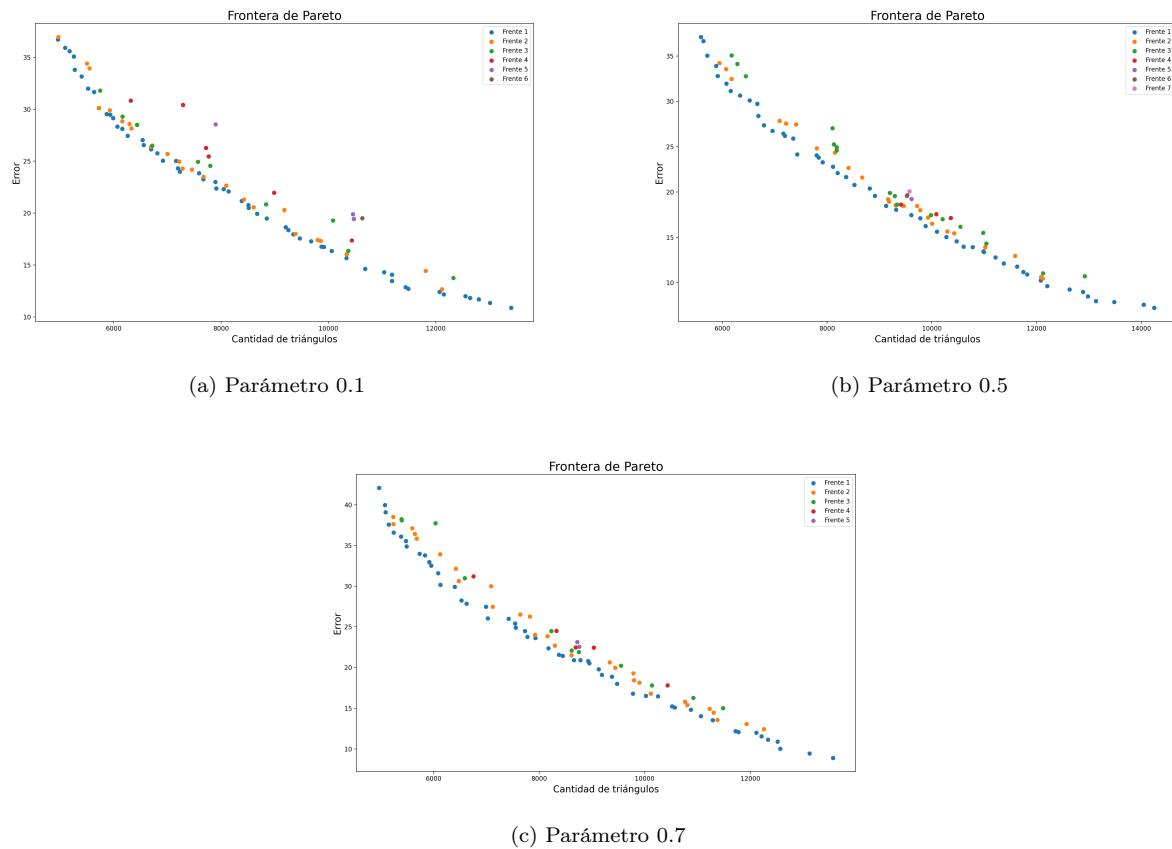
De la tabla anterior se puede concluir que los mejores resultados se presentan cuando el parámetro de cruzamiento es 0.1, ya que este es el que minimiza la cantidad de triángulos y no presenta un error tan elevado. Sin embargo, el términos de la topología de la cara, este presenta deformaciones, por lo que se considera que el parámetro de cruzamiento que tiene mejores resultados es 0.5, pues es el que tiene el menor error y conserva la forma de la cara de la mejor manera.

### Error en los valores de las alturas

Por otra parte, al medir el error como la suma de las diferencias absolutas de la altura real con la estimada, se obtienen los siguientes resultados para las variaciones del parámetro de cruzamiento:



Ahora, las fronteras de Pareto obtenidas son:



De las gráficas anteriores se puede concluir que a medida que aumenta el parámetro de cruzamiento, la distancia entre los frentes formados también incrementa. Esto se debe a que cuando el parámetro de cruzamiento se aleja de 0.5, las nuevas generaciones no varían significativamente respecto a sus ancestros, ya que toman la mayoría de sus

atributos de uno de los padres.

| Parámetro de cruzamiento | No. vértices | No. triángulos | Error   | Tiempo de ejecución | Peso en memoria (KB) |
|--------------------------|--------------|----------------|---------|---------------------|----------------------|
| 0.1                      | 6829         | 13401          | 10.8649 | 14:19.98            | 426                  |
| 0.5                      | 7259         | 14243          | 7.2244  | 14:54.27            | 454                  |
| 0.7                      | 6916         | 13566          | 8.8872  | 14:55.48            | 432                  |

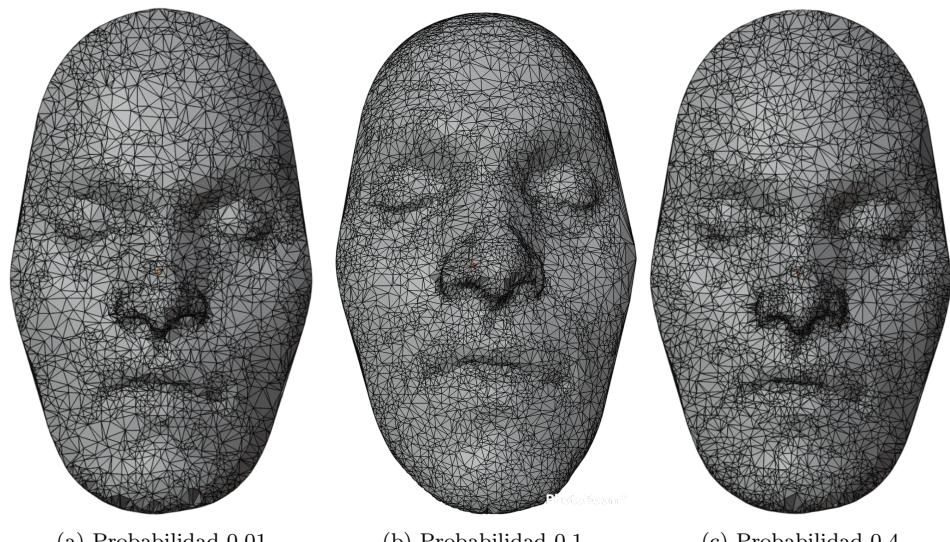
De la tabla anterior, a pesar de que cuando los parámetros de mutación son 0.1 y 0.7 se tiene una menor cantidad de triángulos, estos presentan un mayor error. Además, al ver los modelos obtenidos con estos parámetros, se puede notar una gran pérdida de detalle con respecto a la malla original. Por tanto, se determina que el parámetro de cruzamiento que ofrece mejores resultados es 0.5, ya que este conserva la topología de la cara, lo cual es prioritario en la simplificación de la malla 3D.

### 5.3. Modificación en la probabilidad de mutación

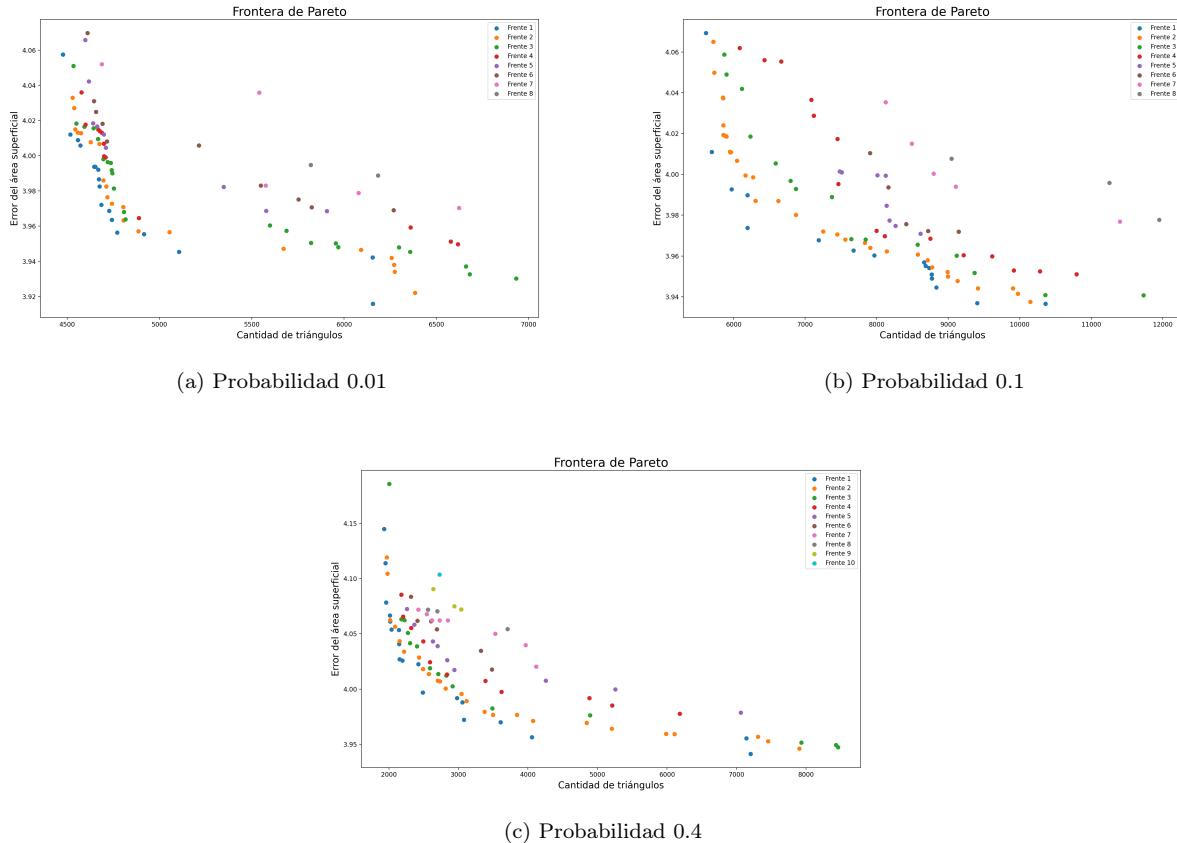
Con el objetivo de analizar cómo cambian los resultados del algoritmo NSGA-II al variar la probabilidad de mutación, esta se cambia a 0.01, 0.1 y 0.4, teniendo en cuenta que este valor debe ser menor al parámetro de cruzamiento. Así pues, para obtener los resultados se tomaron 100 generaciones de 50 individuos con un parámetro de cruzamiento de 0.5.

#### Error en el área superficial

En primer lugar, considerando como métrica del error de la diferencia absoluta en las áreas superficiales, las mallas obtenidas con las diferentes probabilidades de mutación son las siguientes:



Así, las fronteras de Pareto para estos valores son:



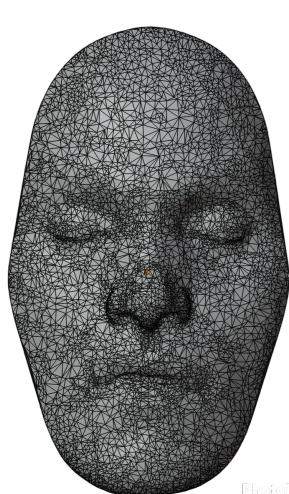
De las gráficas anteriores, se puede ver que hay poca diversidad en las soluciones del primer frente cuando la probabilidad de mutación es de 0.01. Además, en una sección de la gráfica las distancias entre los individuos de los frentes están muy próximos, mientras que en la otra parte son muy dispersas. Por otra parte, cuando la probabilidad de mutación es 0.4, las distancias que hay entre los diferentes frentes son muy pocas, haciendo que haya poca variedad en las soluciones entre un frente y otro. Finalmente, cuando la probabilidad de mutación es 0.1, se distinguen claramente los frentes y hay más variedad en las soluciones. La siguiente tabla contiene las soluciones con la mayor distancia de apilamiento en el primer frente para las diferentes probabilidades de mutación:

| Probabilidad de mutación | No. vértices | No. triángulos | Error  | Tiempo de ejecución | Peso en memoria (KB) |
|--------------------------|--------------|----------------|--------|---------------------|----------------------|
| 0.01                     | 3145         | 6156           | 3.9158 | 05:15.61            | 193                  |
| 0.1                      | 5870         | 11506          | 3.9359 | 06:06.84            | 366                  |
| 0.4                      | 3682         | 7204           | 3.9412 | 04:59.30            | 227                  |

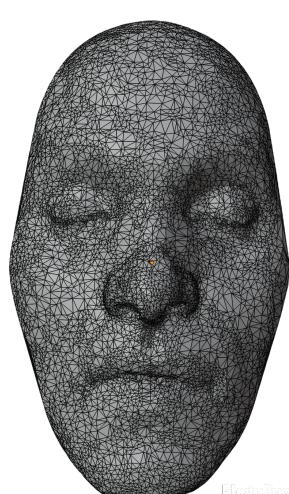
De la tabla anterior es posible notar que cuando la probabilidad de mutación es 0.01, se obtiene tanto el mínimo de vértices como del error. Adicionalmente, con este valor, se conserva la topología de la malla original. Asimismo, cuando la probabilidad de mutación es de 0.1 y 0.4 también se logran buenos resultados en cuanto a la forma de la malla, pero el error y la cantidad de triángulos es mucho mayor a comparación de cuando es de 0.01.

## Error en los valores de las alturas

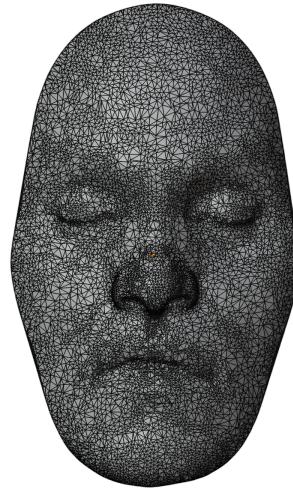
Al utilizar las funciones objetivo de cantidad de triángulos y la suma de las distancias absolutas entre la altura real con la altura estimada, se obtuvieron las siguientes configuraciones de la malla con variaciones de la probabilidad de mutación.



(a) Probabilidad 0.01

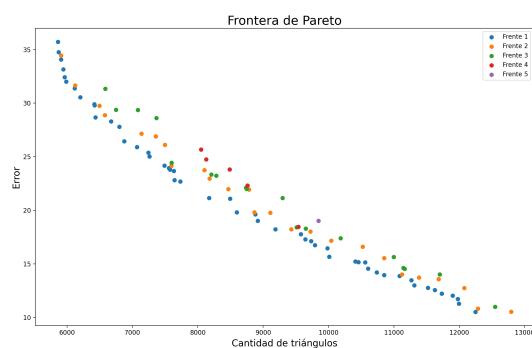


(b) Probabilidad 0.1

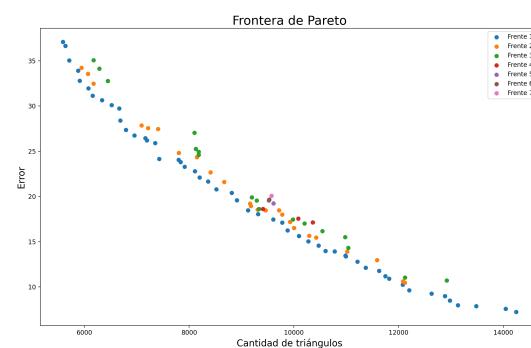


(c) Probabilidad 0.4

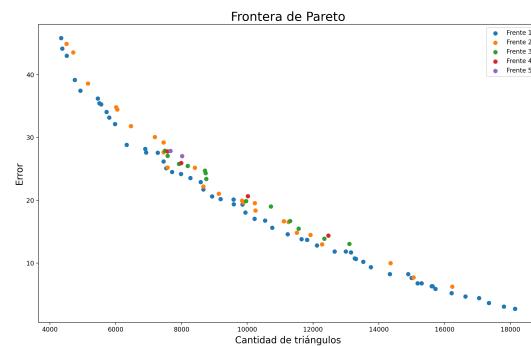
Las fronteras de Pareto obtenidas son:



(a) Probabilidad 0.01



(b) Probabilidad 0.1



(c) Probabilidad 0.4

De las fronteras de Pareto anteriores, se puede ver que cuando la probabilidad de mutación es de 0.01, hay una mejor división entre los frentes, a comparación de los otros dos casos, donde las soluciones entre frente y frente son muy similares.

| Probabilidad de mutación | No. vértices | No. triángulos | Error   | Tiempo de ejecución | Peso en memoria (KB) |
|--------------------------|--------------|----------------|---------|---------------------|----------------------|
| 0.01                     | 6232         | 12246          | 10.5129 | 14:30.95            | 389                  |
| 0.1                      | 7259         | 14243          | 7.2244  | 14:54.27            | 454                  |
| 0.4                      | 9241         | 18138          | 2.6880  | 14:17.84            | 579                  |

De la tabla anterior se puede concluir que cuando la probabilidad de mutación es de 0.4, el error es muy pequeño, pero la reducción en la cantidad de triángulos es casi nula. Por otra parte, cuando este valor es de 0.01, se obtiene la menor cantidad de triángulos, a pesar de que el error es el más elevado. Además, esta probabilidad conserva la topología de la malla, por lo que se considera que es el más indicado para este algoritmo NSGA-II.

## 6. Conclusiones

En primer lugar, utilizar la aproximación multiobjetivo para resolver el problema de simplificación de mallas 3D, mediante la adaptación del algoritmo genético NSGA-II, permite encontrar diversas soluciones con diferentes compensaciones entre la precisión y la simplicidad del modelo 3D asociado a rostros. Por lo que se puede concluir que resolver el problema mediante este método de algoritmos genéticos es una ventaja, ya que no condiciona la solución a ser una sola, sino que se tiene diferentes alternativas solución según lo que se quiera, en este caso, se prefiere una solución con un poco más de precisión que de simplicidad, ya que uno de los interés es conservar lo más que se pueda la topología del rostro.

Por otro lado, de los análisis y resultados obtenidos se puede evidenciar que, cada uno de los parámetros del algoritmo, como la probabilidad de mutación, el parámetro de cruzamiento, la cantidad de generaciones, son factores que tienen un gran impacto en los resultados, por lo que, es de vital importancia escoger los valores óptimos para cada parámetro de tal manera que se puedan obtener los resultados esperados. Dicho esto, se puede concluir que, para que los resultados del algoritmo NSGA-II implementado sean los esperados el número de generaciones que se debe seleccionar debe ser igual a 100, ya que con este se obtiene el menor error posible entre las demás generaciones estudiadas. De la misma manera, el parámetro de cruzamiento adecuado es 0.5, ya que es aquel que permite conservar en gran medida la topología de la malla asociada al rostro. Y finalmente, la probabilidad de mutación óptima para cada individuo es de 0.01.

Con respecto a las dos métricas del error utilizadas, se puede concluir que para la malla utilizada, la mejor solución se obtiene utilizando el error del área superficial, ya que esta logra obtener menos triángulos, conservando la topología de la cara. Adicionalmente, esta métrica tiene un tiempo de ejecución mucho más corto, lo cual proporciona beneficios computacionales a la hora de trabajar con mallas que tienen una gran cantidad de vértices.

## Referencias

- [1] B. R. Campomanes-Álvarez, O. Cordón, and S. Damas, “Evolutionary multi-objective optimization for mesh simplification of 3D open models,” *Integrated Computer-Aided Engineering*, vol. 20, no. 4, pp. 375–390, 2013.
- [2] E. W. Weisstein, *Triangulation*, From MathWorld—A Wolfram Web Resource. [En línea]. Disponible en: <https://mathworld.wolfram.com/Triangulation.html>. [Accedido: 03 de noviembre de 2022].
- [3] M. M. Reparaz and N. A. Rodríguez, “Triangulaciones de Delaunay de alto orden en el terreno práctico de los sistemas de información geográfica,” tesis, 2014.
- [4] E. W. Weisstein, *Delaunay Triangulation*, From MathWorld—A Wolfram Web Resource [En línea]. Disponible en: <https://mathworld.wolfram.com/DelaunayTriangulation.html>. [Accedido: 03 de noviembre de 2022].
- [5] Efel, *Voronoi site points from Delaunay triangulation*, Stack overflow [En línea]. Disponible en: <https://stackoverflow.com/questions/42047077/voronoi-site-points-from-delaunay-triangulation>. [Accedido: 03 de noviembre de 2022].
- [6] D. E. Goldberg, *Genetic algorithms in machine learning, search and optimization*. Addison- Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.
- [7] “What Is the Genetic Algorithm?”, MathWorks [En línea]. Disponible en: <https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html> [Accedido: 04 de noviembre de 2022].
- [8] C. Zavoianu. “Enhanced Evolutionary Algorithms for Solving Computationally-Intensive Multi-Objective Optimization Problems”, tesis, Johannes Kepler University, Austria, Linz, 2015.
- [9] V. De Bucka, C. Muñoz, P. Nimmemeers, I. Hashema, y J. Impe. “Multi-objective optimisation of chemical processes via improved genetic algorithms: A novel trade-off and termination criterion”. *Computer Aided Chemical Engineering*, vol 46, pp 613-618, 2019.
- [10] A. Ouni, K. Kessentini y H. Sahraoui. “Multiobjective Optimization for Software Refactoring and Evolution”. *Advances in Computers*, vol 94, pp 103-167, 2014.
- [11] NPTEL IIT Guwahati, *Lec 21 : Non-Dominated Genetic Algorithm: NSGA-II: Introduction*. (Feb 24, 2021). Accedido: 04 de octubre de 2022. [Video Online]. Disponible en: <https://www.youtube.com/watch?v=Aq4pwGn5uWY>
- [12] Deb. K, Pratap. A, Agarwal. S y . Meyarivan. T. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”. *IEEE Transactions on evolutionary computation*, Vol. 6, No. 2, 2002.
- [13] “Lecture 9: Multi-objective Optimization.” [En línea]. Disponible en: <https://engineering.purdue.edu/~sudhoff/ee630/Lecture09.pdf>. [Accedido: 04 de octubre de 2022].

- [14] A. Seshadri, “A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II” [En línea]. Disponible en: [https://web.njit.edu/~horacio/Math451H/download/Seshadri\\_NSGA-II.pdf](https://web.njit.edu/~horacio/Math451H/download/Seshadri_NSGA-II.pdf) [Accedido: 04 de octubre de 2022].
- [15] Yeggi - 3D Printer Models Search Engine [En línea]. Disponible en: <https://www.yeggi.com>