# Algoritmo genético multiobjetivo NSGA-II para la simplificación de mallas 3D

Cristian Alzate Urrea
*Ingeniería Matemática,*
*Escuela de Ciencias aplicadas*
*e ingeniería,*
*Universidad EAFIT,*
*Medellín, Colombia*

Martín Sanchez Reyes
*Ingeniería Matemática,*
*Escuela de Ciencias aplicadas*
*e ingeniería,*
*Universidad EAFIT,*
*Medellín, Colombia*

***Abstract*—Models of surfaces represented in 3d grids are used in various contexts, such as design of new products, spaces, geographic data processing and game devlopment [1].**

**In general, the processing of this models demand a high comptational effort, which result in less user accessibility in lower capacity hardware. For this reason, it becomes necessary to simplify this models reducing the points forming them and so reducing the number of triangles used to represent the surface.**

**This reduction must be done causing the least damage to the quality of the oringial mesh. This present a multiobjective optimization problem. As an approach, we use in this work the non sorted genetic algorithm NSGA II, which is well suited for the problem statement. We will detail the experimentation made over the paremeters of combinations, number of generations required, mutation rates and size of population chosen. Later we comment on the metodology and implementation of the algorithm, we show aswell the resulting meshes and comment on the performance of the NSGA-II.**

***Keywords*—Mulobjective Optimization, NSGA-II, 3D surfaces, 3D meshes, Pareto optimality.**

## 1. Introducción

Los modelos de superficies representadas en mallas 3d se usan en diversos contextos, como el diseño de nuevos productos, ambientación de espacios, manejo de datos geográficos o la elaboración de videojuegos [1].

El procesamiento de estos modelos generalmente implican un costo computacional muy alto, lo que resulta en poca extensibilidad de uso en distintos tipos de hardware de menor capacidad. Por dicha razón, se hace necesario simplificar los modelos, reduciendo el número de puntos que los forman y por tanto los triángulos usados para representar la superficie.

Esta reducción debe hacerse con el menor deterioro en la calidad de la malla original. La aproximación escodiga es la optimización multiobjetivo. Esto presenta un problema de optimización multiobjetivo. Se aborda la simplificación con el algoritmo genético de clasificación elitista no dominante NSGA-II, por sus siglas en inglés, que se adapta bastante bien al tipo de planteamiento elegido. Se detallará la experimentación que se hizo sobre parámetros del algoritmo como son las combinaciones, el número de generaciones requeridas, tasas de mutación y número de individuos en la población. Posteriormente se comenta la metodología y la implementación del algoritmo en el problema específico, incluyendo los resultados en donde se observarán las nuevas mallas y el desempeño del NSGA-II.

## 2. NSGA-II

El algoritmo genético de clasificación no dominante NSGA fue propuesto por Srinivas y Debes en [2]. Luego, para mejorar problemas de complejidad algorítmica, en [3] se presenta un algoritmo elitista que aborda estos problemas, planteando el NSGA-II. Como todo algoritmo genético, tiene una inspiración biológica detrás de su funcionamiento. El NSGA-II parte de una población inicial. La malla original tendrá un tamaño N fijo. Se seleccionará inicialmente un conjunto de puntos, cada individuo de la población está codificado por un cromosoma (vector del mismo tamaño N) binario. Cada individuo será una configuración simplificada de la malla original, donde sus entradas codifican los puntos incluidos (1) o excluidos (con 0). Cada individuo es evaluado bajo una función de aptitud en cada objetivo del problema de optimización multiobjetivo. Sobre esta evaluación se genera un orden de clasificación bajo la dominancia de Pareto. Dependiendo del número de individuos por los que es dominado, un punto pertenece a una ”frontera de Pareto”. A partir de la recombinación de los individuos, se genera una generación nueva, mezclando aleatoriamente dos padres de la población de la generación actual aplicando un crossover uniforme. El criterio de terminación elegido fue predefinir un número específico de generaciones.

## 3. Metodología

El proceso metodológico se resume en la proyección de los puntos tridimensionales de la malla al plano $xy$. La población es inicializada usando los puntos proyectados. Sobre éstos se genera una triangulación de Delaunay, la cual

maximiza los ángulos internos de los triángulos formados entre los vértices. Los dos objetivos a minimizar del modelo será el error, que calculamos como la diferencia de áreas entre las dos mallas, y la complejidad de la malla, indicada en el número de puntos que contiene. De esta forma, siendo $M_0$ la malla original, formada por k triángulos $T_i$ con correspondiente área $A(T_i)$ y, análogamente $m$ el cromosoma de una malla individuo con triángulos $t_i$ de área $A(t_i$, se plantea la minimización de:

$$minE(m) = \sum_{i}^{k} A(T_i) - A(t_i) \quad (1)$$

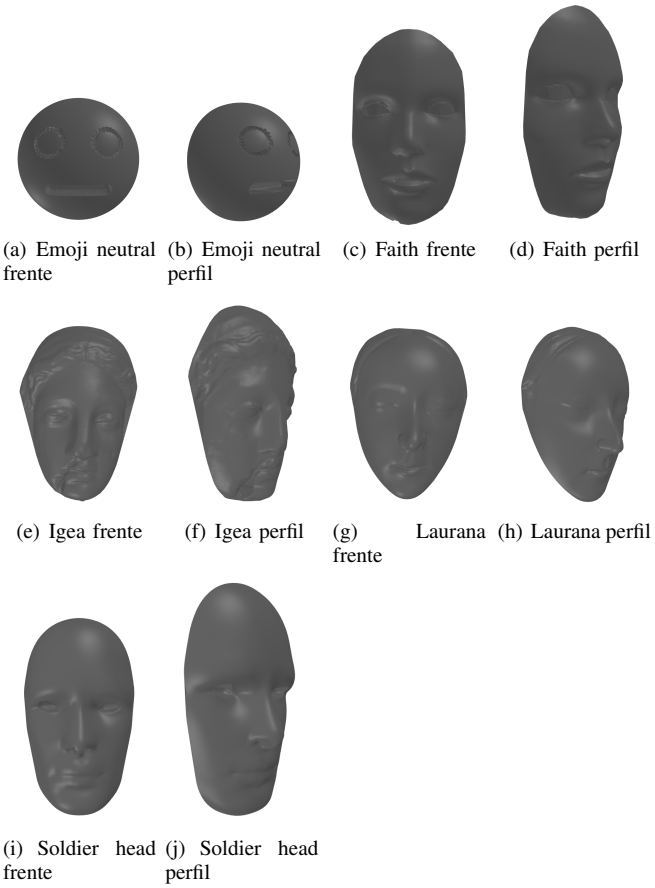
en simultáneo con:

$$minC(m) = \sum_{j}^{N} m_j \quad (2)$$

# 4. Experimentos

Para probar el desempeño del algoritmo utilizamos 5 mallas: Emoji_Neutral.obj, Faith.obj, Igea.obj, Laurana.obj y soldier_head.obj, sacados de [4]. Todas son mallas con formato obj, que básicamente permite almacenar las coordenadas de puntos, así como colores y texturas de un objeto.

A continuación se muestra cada uno de los datasets:



(a) Emoji neutral frente (b) Emoji neutral perfil (c) Faith frente (d) Faith perfil

(e) Igea frente (f) Igea perfil (g) Laurana frente (h) Laurana perfil

(i) Soldier head frente (j) Soldier head perfil

Figure 1. Mallas de prueba.

## 4.1. Preparación experimental

El algoritmo genético NSGA-II ha sido implementado en el lenguaje Python, y se probó en un procesador AMD Ryzen 5 3500U con Radeon Vega Mobile Gfx (8) 2.100GHz, y 12 Gb de RAM, usando el sistema operativo Arch Linux x86_64.

Se escogió Python por ser un lenguaje sencillo y rápido para el desarrollo, además de poseer gran cantidad de librerías que facilitan la manipulación de mallas 3d.

Para realizar la recombinación utilizamos crossover uniforme, por su simplicidad y porque generaba resultados suficientemente buenos, es decir, cada generación presentaba una mejora respecto a la anterior. Los parámetros establecidos para el algoritmo se tomaron y probaron de [5], encontrando que son los mejores dentro de los que se consideraron empíricamente:

- Tamaño de la población: 100
- Generaciones: 50
- Probabilidad de crossover: 0.8
- Probabilidad de mutación: 1/longitud_cromosoma

Las poblaciones iniciales son generadas aleatoriamente a partir de la malla original. Para ello se elimina una cantidad aleatoria de puntos en la malla siendo la malla resultante un miembro de la población inicial.

Además, para la triangulación de Delaunay que no permitirá reconstruir las mallas usamos la librería Scipy de python, y para graficar las mallas usamos Trimesh, también de Python. En cuanto al ordenamiento no dominado y el cálculo de la distancia crowding se usaron los algoritmos descritos en [3], ya que son más rápidos y eficientes en tiempo, sacrificando memoria.

Asimismo, la primera función objetivo, el error, se midió la diferencia entre el área de la superficie del individuo observado y el área de la superficie de la malla original. Por otro lado, para la segunda función objetivo se contabilizó el número de puntos que poseía cada individuo.

## 4.2. Comparación de rendimiento: frontera de Pareto

En muchos problemas de la realidad se podrá hallar la frontera de Pareto certera para comparar las soluciones encontradas respecto a las mejores. Sin embargo, dadas las características de este problema no podremos hacerlo, pero usamos una forma para ir observando las mejores soluciones a lo largo de las generaciones.

Para ello nos aprovechamos del ordenamiento no dominado, y graficamos todos los individuos destacando los de la primera frontera, siendo estos los que pertenecen a nuestra frontera de Pareto para dicha generación.

## 4.3. Métrica de rendimiento: indicador de calidad

Los indicadores de calidad son importantes para escoger entre las diferentes opciones que tenemos en al frontera de

Pareto, ya que con esta podremos decidir a qué criterios les daremos mayor importancia para nuestra solución. En nuestro caso le daremos mayor importancia al error, por lo que usaremos la siguiente fórmula para medir el rendimiento de cada individuo:

$$P = \sum_{i=0}^{n} \lambda_i \frac{O_n}{n} \tag{3}$$

donde $n$ es el número de funciones objetivos que tenemos, $O_n$ es el valor del objetivo n-ésimo, y $\lambda_i \in [0, 1]$. Además:

$$\sum_{i=1}^{n} \lambda_i = 1$$

Mediante esta métrica escogeremos la mejor malla generada.

## 5. Resultados

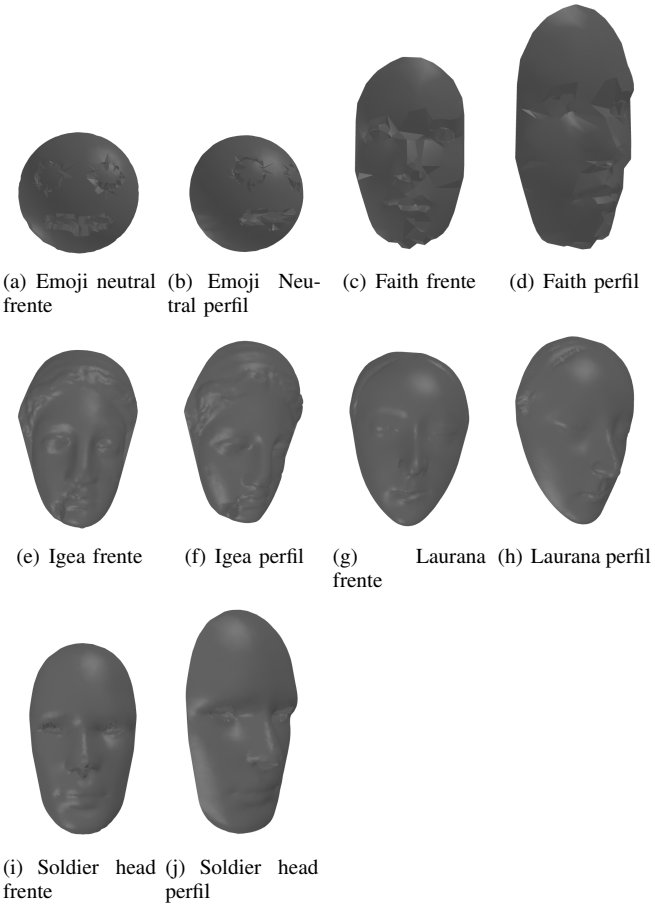
A continuación mostraremos los resultados de haber probado el algoritmo para reducir puntos a las mallas 3d. En la figura 2 se observa la evolución de los valores en las funciones objetivo para el dataset Laurana.obj a través de las generaciones, en el que solo se muestran las generaciones con potencias de 10. Allí se evidencia la mejora de las generaciones con el paso de las iteraciones, ya que nuestro objetivo es de minimización, por lo tanto entre más se ajuste la frontera a los ejes ordenados será mejor. Un resultado similar se obtiene con el resto de mallas de prueba.

Además, en la figura 3 se puede observar la evolución de las fronteras de Pareto a través de las generaciones para cada uno de los datasets, incluyendo Laurana.obj; allíse evidencia que las soluciones van mejorando con el paso del tiempo, ya que se van ajustando a los ejes ordenados, es decir, van minimizando las funciones objetivo.

Así pues, en el gráfico 4 se observan la fronteras de Pareto de la última generación para cada uno de los datasets probados.

A partir de dichas fronteras de Pareto se determinaron las mallas que mejor se adaptaban a la solución del problema con la métrica descrita en la sección 4.3. Por lo tanto, en la tabla 1 vemos las soluciones escogidas como nuestras mallas finales para cada uno de los datasets.

Las mallas que tuvieron mayor reducción fueron Faith y Emoji neutral, mientras que las otras tres tuvieron un desempeño similar. Finalmente, la vista obtenida de las mallas tras su reducción se aprecia en la figura 5.



(a) Emoji neutral frente (b) Emoji Neutral perfil (c) Faith frente (d) Faith perfil

(e) Igea frente (f) Igea perfil (g) Laurana frente (h) Laurana perfil

(i) Soldier head frente (j) Soldier head perfil

Figure 5. Mallas de prueba reducidas.

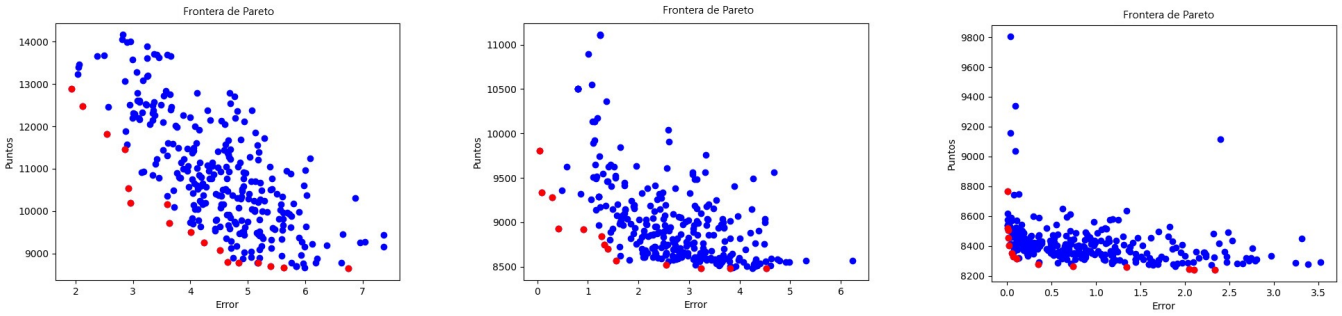
### 5.1. Inconvenientes con el método

Como se mencionó anteriormente, unas de las condiciones para este problema es que las superficies sean funciones, ya que la triangulación de Delaunay no funciona bien en otro caso. Sin embargo, es de resaltar que ninguna superficie cumplía completamente este requisito, por lo que hay desórdenes y zonas mal suavizadas en algunas mallas finales. Este fenómeno se nota más en algunas que en otras.

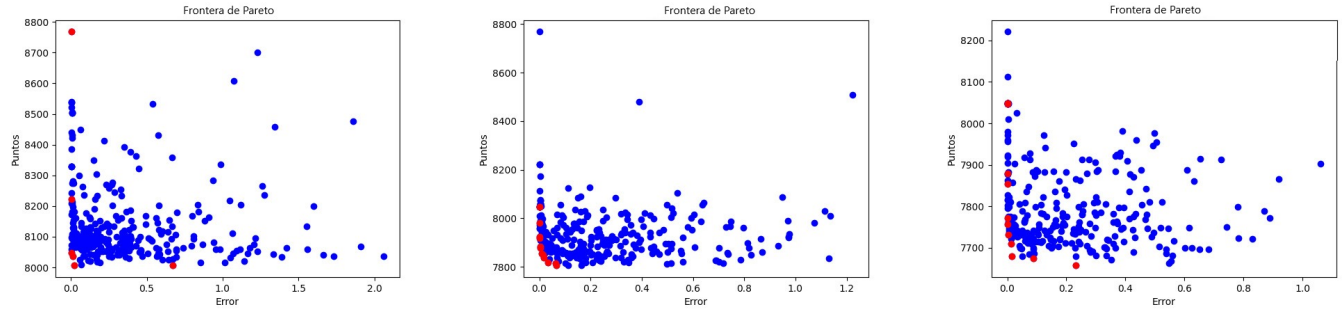
Una forma en la que se puede atacar este problema es utilizando otras formas de triangulación, sobre todo las que están enfocadas a la triangulación espacial, como por ejemplo la triangulación alpha-shape [6].

## 6. Conclusiones

El NSGA-II es un buen método para reducir la complejidad y los puntos de las mallas 3d. Esto se pudo comprobar con los resultados que obtuvimos durante la experimentación. Sin embargo, es de resaltar que el método de triangulación puede ser mejor, ya que nos exige bastante con tener superficies en forma de función. Esto no es siempre posible en la práctica, de hecho, es poco común. Por lo tanto, se podría mejorar el método utilizando otros tipos de triangulación, incluso enfocadas a la triangulación espacial.

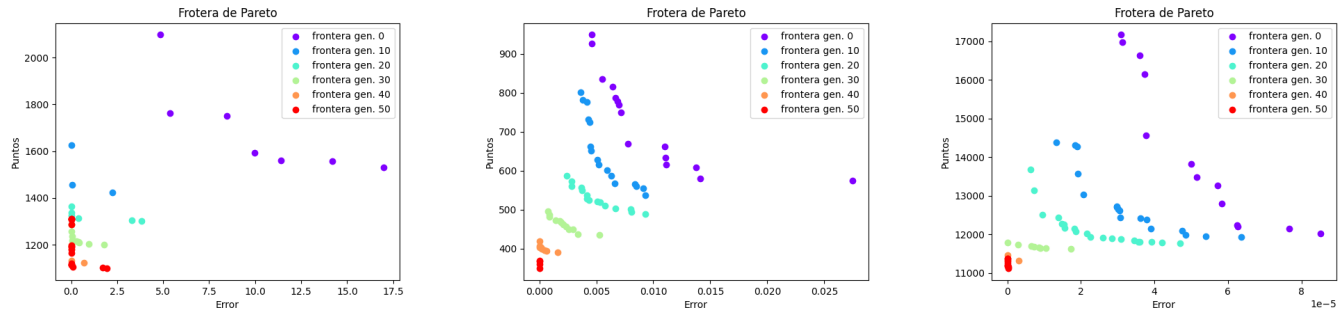



(a) Generación 0 (b) Generación 10 (c) Generación 20

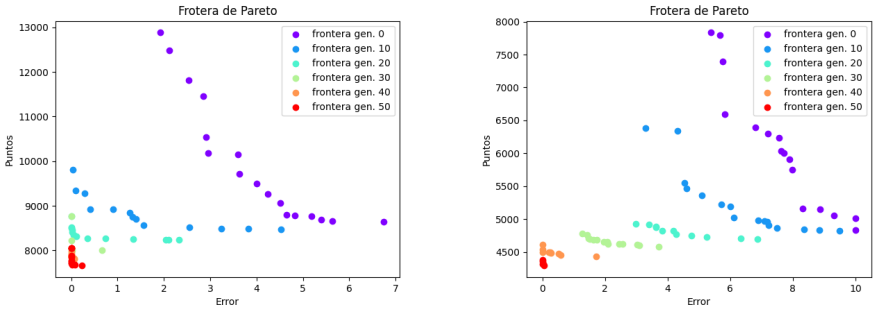



(d) Generación 30 (e) Generación 40 (f) Generación 50

Figure 2. Fronteras de Pareto para cada generación de potencia de 10 en el dataset Laurana.



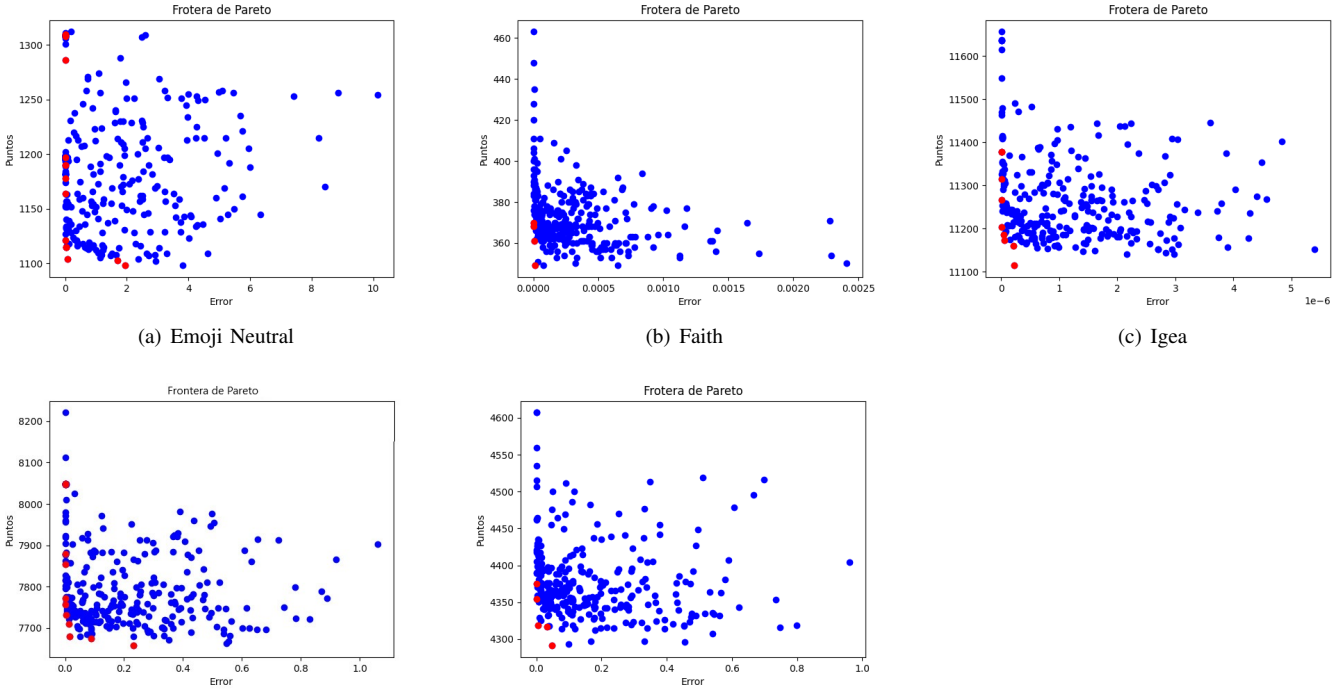

(a) Emoji Neutral (b) Faith (c) Igea




(d) Laurana (e) Soldier head

Figure 3. Evolución Fronteras de Pareto en generaciones potencias de 10

Además, un punto a mejorar es la forma de medir el error, ya que la diferencia entre el área de las superficies no




(a) Emoji Neutral (b) Faith (c) Igea

(d) Laurana (e) Soldier head

Figure 4. Fronteras de Pareto de la última generación para cada dataset.

| Dataset | Error | Numero puntos | Numero puntos originales | Reducción puntos | Porcentaje reducción |
|---|---|---|---|---|---|
| **Emoji neutral** | 1.945 | 1098 | 4160 | 3062 | 73.6% |
| **Faith** | 1.475 | 349 | 1594 | 1245 | 78.1% |
| **Igea** | 2.236 | 11114 | 32216 | 21147 | 65.54% |
| **Laurana** | 0.231 | 7657 | 23395 | 15738 | 67.27% |
| **Soldier head** | 0.049 | 4291 | 13264 | 8973 | 67.64% |

TABLE 1. MALLAS SELECCIONADAS DE LA FRONTERA DE PARETO

es tan precisa como debería, porque por ejemplo se podría obtener un triángulo con el área igual a la de la superficie inicial, pero definitivamente no habría similitud entre ambas mallas.

## References


[1] S.-Y. H. Hui-Ling Huang, "Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm," *Pattern Recognition*, vol. 36, p. 1065 – 1081, 2003.

[2] N. Srinivas and K. Deb, "Multi-objective function optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[3] A. P. Kalyanmoy Deb, Samir Agrawal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *E TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 6, no. 2, pp. 182–197, 2002.

[4] Turbosquid, "Modelos en 3d para profesionales," 2022. [Online]. Available: https://www.turbosquid.com/es/

[5] B. R. Campomanes-Álvarez, O. Cordon, 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.

[6] M. Bern, J. R. Shewchuk, and N. Amenta, "Triangulations and mesh generation," in *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017, pp. 763–785.