INSTITUTO TECNOLÓGICO DE COSTA RICA

ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA

MT 8008
Inteligencia Artificial

Tarea 2 – Parte 2

Computación Evolutiva: Optimización Multiobjetivo

Jose Fabio Navarro Naranjo – 2019049626

César Argüello Salas – 2019047699

Profesor: Juan Luis Crespo Mariño

Semestre II - 2022

Análisis del problema

i. *Fundamento Teórico*

El problema planteado trata sobre la determinación de la posición de 3 pernos, los cuales tiene como función, sostener una viga que se encuentra colocada en voladizo en una columna. Dicha viga debe soportar un carga de 2000 lb, la cual se efectúa de manera vertical como se muestra en la figura 1.

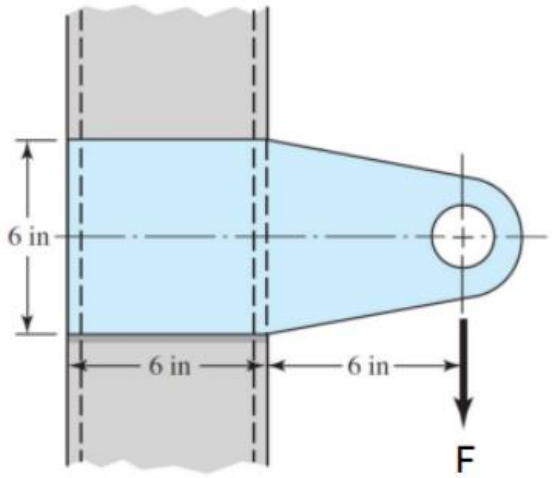

Figura 1. Problema planteado.

La finalidad del problema es determinar la posición donde dichos pernos se sometan a la menor fuerza de modo que éstos tengan un costo menor y se pueda mantener la funcionalidad del sistema. Por lo anterior, para poder llevar a cabo la solución del problema, primero se debe realizar el estudio estático del sistema. Es importante mencionar que, para dicha solución, se recomienda una distribución triangular de los pernos.

Debido a la configuración del sistema, la fuerza efectuada en cada perno tendrá 2 componentes, una debida a la fuerza cortante (producto directo de la fuerza de 2000lb efectuada en la viga), y otra debido al momento de flexión (debido a la distancia entre cada perno y el punto de aplicación de la fuerza).

Asimismo, se asume que todos los pernos se encuentran distribuidos en un círculo de radio “r”, de modo que exista la misma distancia entre cada uno de ellos. De igual manera, se considera también que la distribución de los pernos esté desplazada un ángulo “α” medido a partir de la horizontal, como se muestra en la figura 2. Lo anterior se realiza con el fin definir la posición de los 3 pernos con únicamente 2 parámetros, el radio del círculo donde se encuentran ubicados, y el ángulo de desplazamiento que tiene la configuración, ya que como se encuentran igualmente espaciados, se sabe que existen 120° entre cada uno de ellos.

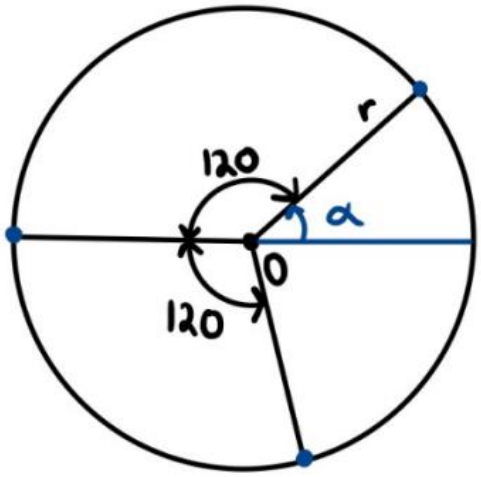

Figura 2. Distribución de los pernos.

Gracias al análisis anterior, la magnitud de las fuerzas que afectan a cada uno de los pernos se detalla en las ecuaciones (1), (2), y (3) que se muestran a continuación.

$$|F_{t1}| = \frac{F}{3}\sqrt{\left(\frac{9}{r}sen(\alpha)\right)^2 + \left(1 + \frac{9}{r}\cos(\alpha)\right)^2} \quad (1)$$

$$|F_{t2}| = \frac{F}{3}\sqrt{\left(\frac{9}{r}sen(\alpha + 120)\right)^2 + \left(1 + \frac{9}{r}\cos(\alpha + 120)\right)^2} \quad (2)$$

$$|F_{t3}| = \frac{F}{3}\sqrt{\left(\frac{9}{r}sen(\alpha + 240)\right)^2 + \left(1 + \frac{9}{r}\cos(\alpha + 240)\right)^2} \quad (3)$$

Debido al análisis anterior, se plantea que, para la solución de dicho problema, se puede implementar un algoritmo de computación evolutiva, específicamente una optimización multiobjetivo (MOO por su siglas en inglés), donde las 3 funciones a optimizar son las funciones que se muestran en las ecuaciones (1), (2), y (3), por lo cual dichas ecuaciones van a ser la denominadas funciones de calidad del algoritmo evolutivo.

Como se puede observar, dichas funciones dependen de los parámetros “r” y “α”, los cuales se van a usar como los genes del algoritmo. Es importante mencionar que los valores del ángulo “α” se encuentran en el rango de 0° a 180°. Asimismo, tomando como base un perno de 3/4 in (debido a la gran carga aplicada, se necesita un perno robusto), el valor del parámetro del radio se encuentra aproximadamente entre $\frac{4}{9}\sqrt{3}$ in (aproximadamente 0.77 in), ya que es la distancia desde un vértice (centro del perno) y el centroide de un triángulo (centro del círculo donde se distribuyen los pernos de modo que, en el radio mínimo, los pernos sean tangentes) y $\frac{21}{8}$ in (2.625 in, que corresponde al radio de 3in menos el radio del perno, para que en el caso donde el radio sea el mayor posible, este sea tangente al borde de la superficie donde se va a sujetar).

Descripción de la solución

i. *Codificación*

Como se mencionó anteriormente, para la solución al problema planteado se utilizó un algoritmo de computación evolutiva para lograr una optimización multiobjetivo. El planteamiento y desarrollo de dicho algoritmo se describe a continuación, sin embargo, la codificación completa y documentada se adjunta en los anexos.

Primeramente, para la codificación y solución de dicho problema se utilizó la librería de Python “deap”, debido a ser una herramienta robusta para la codificación de problemas de computación evolutiva. Asimismo, se utilizaron otras librerías como: “random”, “math”, “numpy”, “pandas” y “matplotlib”, para facilitar en si la codificación, el procesamiento de los datos, y la presentación de estos.

Al iniciar la codificación, primero se definieron los “parámetros fijos” o parámetros asociados a la naturaleza del problema, tales como la dimensión o tamaño de los individuos, los límites de los genes e incluso, la fuerza que se aplica a la viga. Luego de esto, se definieron los hiperparámetros, los cuales se estudiarán más adelante, donde se incluyeron todos los aspectos que puede llevar al algoritmo a mejorar o empeorar su comportamiento. Dentro de los hiperparámetros, se consideraron: la cantidad de generaciones, el tamaño de las poblaciones, la cantidad de hijos, la probabilidad de cruzamiento, la probabilidad de mutación y el tamaño de los torneos, para el caso de la selección.

Una vez definidos los parámetros del algoritmo, se continuó creando las clases que iban a soportar por completo el funcionamiento del algoritmo. En estas clases se incluyeron la creación del individuo, y de una calidad de minimización para los 3 objetivos. Así mismo, se registraron el individuo y la población, y los operadores para evaluar la calidad y realizar cruzamiento, mutación, selección y eliminación de individuos.

Específicamente para la creación de individuos, se creó una función que construye listas de 2 valores (uno asociado a cada gen), Dicha función, crea cada uno de los genes de manera aleatoria, siguiendo los límites que se definieron en los parámetros fijos del problema.

De igual manera, para el operador de mutación, se realizó una función que realiza la mutación de los genes, de modo que primero intenta mutar al primer gen (basado en la probabilidad definida en los hiperparámetros), y luego de la misma manera intenta mutar el segundo gen, generando un nuevo valor aleatorio en cada caso.

Asimismo, para el operador de evaluación se desarrolló otra función, la cual alberga las 3 funciones de calidad, o funciones objetivo que el problema tiene (las cuales fueron mencionadas en el planteamiento del problema que se realizó previamente).

Ahora bien, una vez definidos los operadores, y sus respectivas funciones asociadas, se realizó la función principal, la cual se encargar de implementar directamente el procesamiento del algoritmo evolutivo. En esta función principal primero se procede a

inicializar las estadísticas que se van a utilizar para cuantificar el éxito del modelo, las cuales en este caso son el Frente de Pareto, y la desviación estándar entre las calidades de los 3 objetivos. Asimismo, se crea la población inicial. Luego de esto se procede con el inicio de la evolución en el algoritmo.

Primero, se obtiene y se almacena la calidad de toda la población generada. Luego, se continua con el procesamiento y análisis de cada generación, el cuál terminará cuando se generen, procesen y analicen todas las generaciones que se definieron en los parámetros del programa. Durante esta etapa, primero se pasa a través del operador de selección, el cual, mediante el mecanismo de torneo, toma 2 individuos aleatorios, y conserva únicamente el de mejor calidad. Seguidamente, se clonan los resultados del torneo, y se procede a aplicar los operadores de cruzamiento y mutación en los individuos de la población, para finalmente evaluarlos y eliminar en cada iteración al individuo con la peor calidad de toda la población. Esto se realiza para evitar que individuos con calidades realmente bajas puedan afectar el funcionamiento del algoritmo. Luego de esto, se extiende la población resultante, se guardan las estadísticas, y se continúa con la siguiente generación.

Una vez terminado el procesamiento de todas las generaciones del algoritmo, la función principal devuelve las estadísticas (Frente de Pareto, desviación estándar y cantidad de generaciones), para luego enviar estos datos a 3 diferentes funciones. La primera, encargada de graficar la desviación estándar frente a las generaciones, de modo que se puedan utilizar estos datos para un futuro análisis. La segunda para graficar el Frente de Pareto tanto de manera bidimensional (todas las posibles combinaciones entre los objetivos) como tridimensional (todos los objetivos en un solo gráfico). Finalmente, la tercera, se encarga de tabular el Frente de Pareto, los individuos que generan dicho frente, así como la desviación estándar de los componentes del frente.

ii. *Estudio de hiperparámetros y determinación del Frente de Pareto*

Como se mencionó anteriormente, los hiperparámetros presentes para estudio son los siguientes:

- Número de generaciones (NGEN)
- Número de individuos en la población (MU)
- Probabilidad de cruzamiento (CXPB)
- Probabilidad de mutación (MUTPB)
- Número de hijos (NH)
- Tamaño del torneo (TS)

Es importante mencionar que, para dicho estudio, se utilizarán tanto la gráfica de la desviación estándar como la gráficas del Frente de Pareto, con el fin de contrastar ambos resultados, y obtener el mejor grupo de soluciones posibles.

Para comenzar con las iteraciones y el estudio de los hiperparámetros mencionados, se comenzó partiendo de los siguientes valores:

- Número de generaciones: 100

- Número de individuos en la población: 100
- Probabilidad de cruzamiento: 0.8
- Probabilidad de mutación:0.1
- Número de hijos: 50
- Tamaño del torneo: 2

Con dichos valores, se obtuvieron las gráficas que se muestran en la figura 3 y 4.

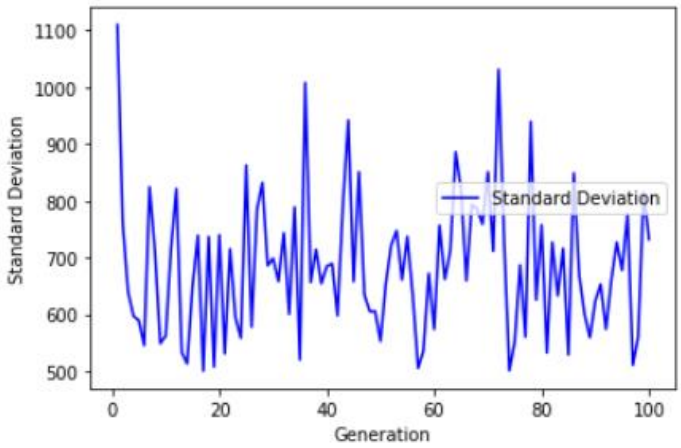

Figura 3. Desviación estándar respecto a las generaciones, iteración 1.

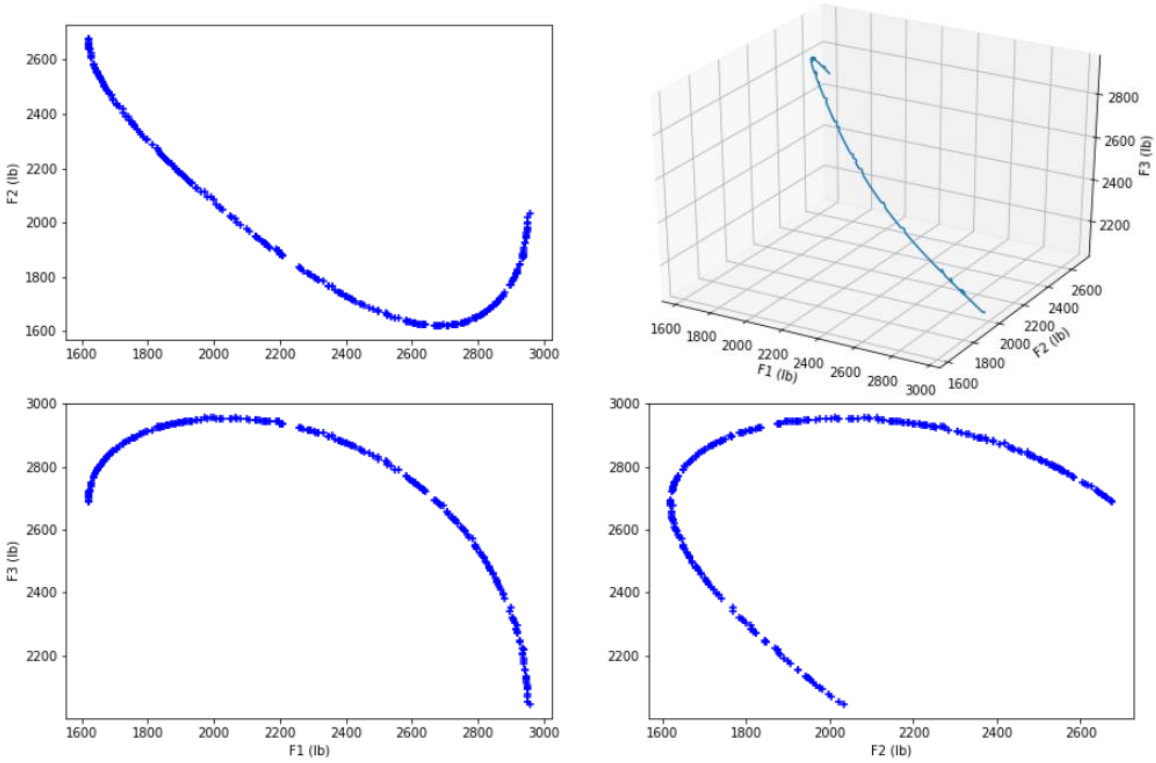

Figura 4. Frente de Pareto, iteración 1.

En esta primer iteración, se observa en el gráfico de la figura 3, que el algoritmo tiende bastante a favoreces la variabilidad y no tanto la presión selectiva, ya que el gráfico se observa bastante variante y poco estable. Por otra parte, el Frente de Pareto se observa bien definido y con una gran cantidad de puntos (306 específicamente para este caso), lo cual se debe a la gran cantidad de generaciones y de individuos en cada población. Por lo cual, para la siguiente iteración se va a reducir abruptamente la cantidad de generaciones, para encontrar el punto óptimo de este hiperparámetro, con el fin de consumir los mínimos recursos computacionales.

En la figura 5, se muestra el Frente de Pareto obtenido con 20 generaciones.

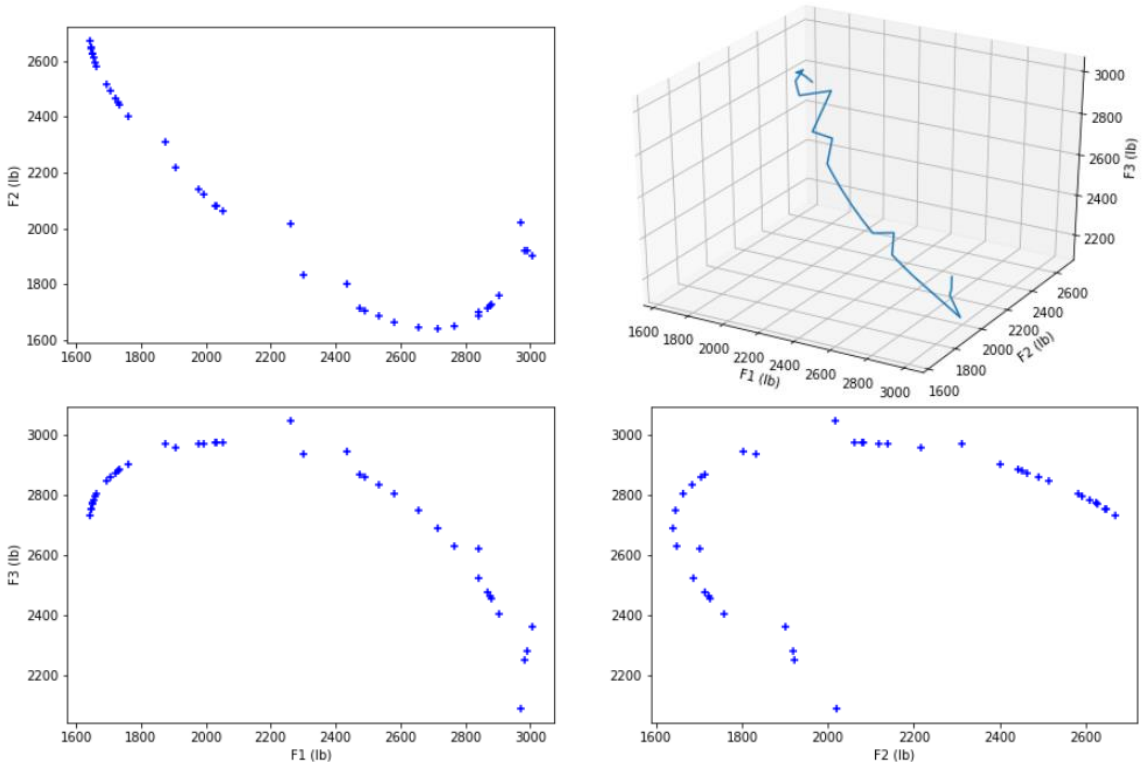

Figura 5. Frente de Pareto, iteración 2.

Como se observa en la figura 5, el Frente de Pareto es bastante deficiente, ya que, debido a la baja cantidad de generaciones, el algoritmo no puede explorar suficientes posibilidades para poder encontrar puntos que se encuentren en el Frente de Pareto.

Debido a lo anterior, en la figura 6 se muestra el Frente de Pareto con un incremento de 10 poblaciones más, es decir 20 en total, para observar el comportamiento del Pareto conforme crecen la cantidad de generaciones.

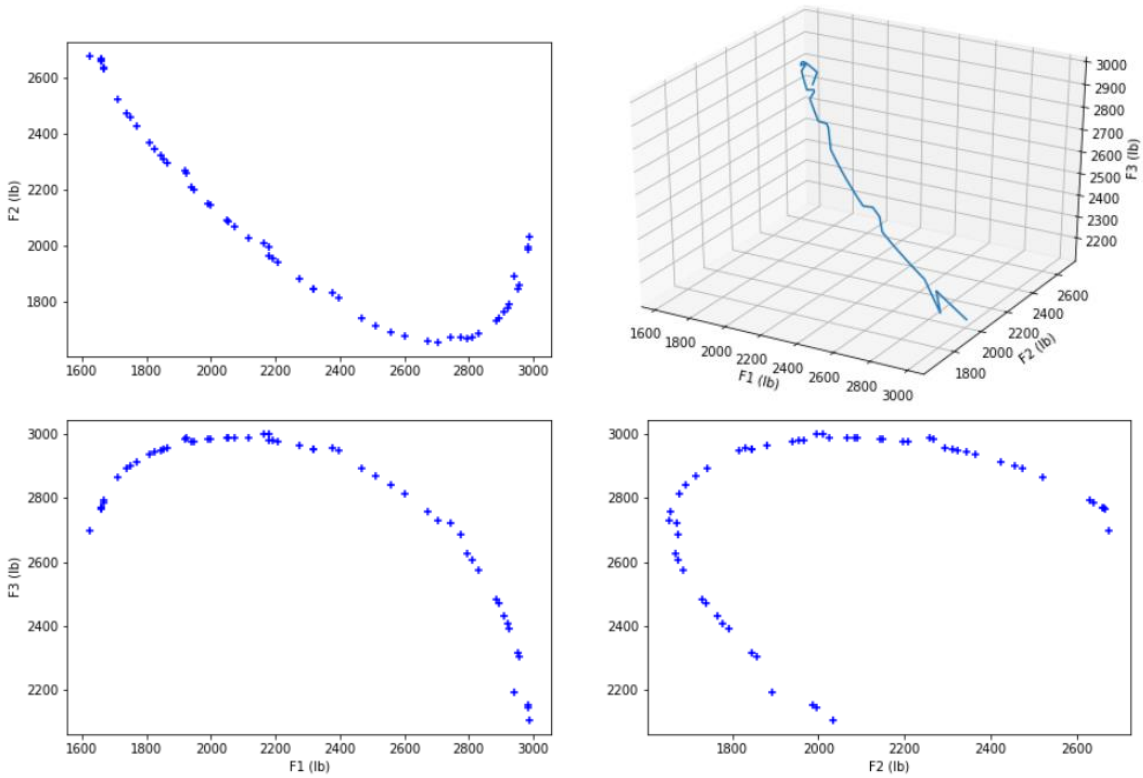

Figura 6. Frente de Pareto, iteración 3.

Como se observa en la figura 6, el gráfico continúa con una deficiencia de puntos, por lo cual se continúa incrementando la cantidad de generaciones. En la figura 7, se muestran los resultados con 40 generaciones.

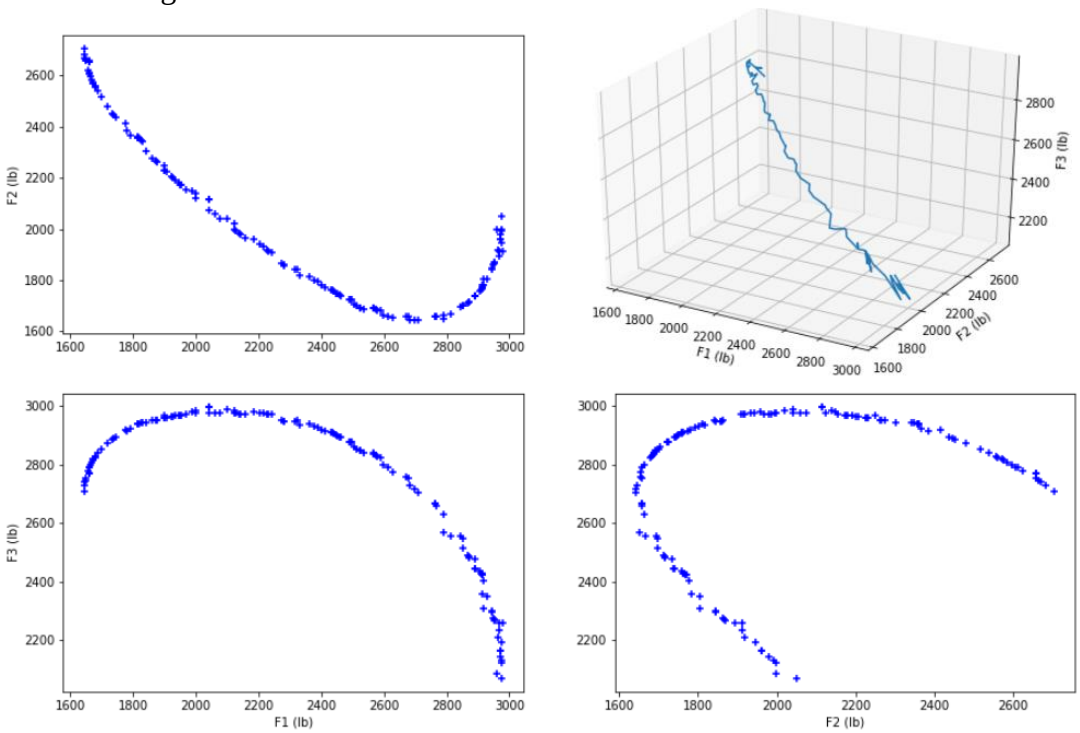

Figura 7. Frente de Pareto, iteración 4.

Acorde con la figura 7, se observa que 40 generaciones aún no son suficientes para generar la cantidad de puntos necesarios en el Frente de Pareto, de modo que este quede bien definido, por lo que, una vez más, se incrementará este valor en 20, alcanzando un total de 60 generaciones.

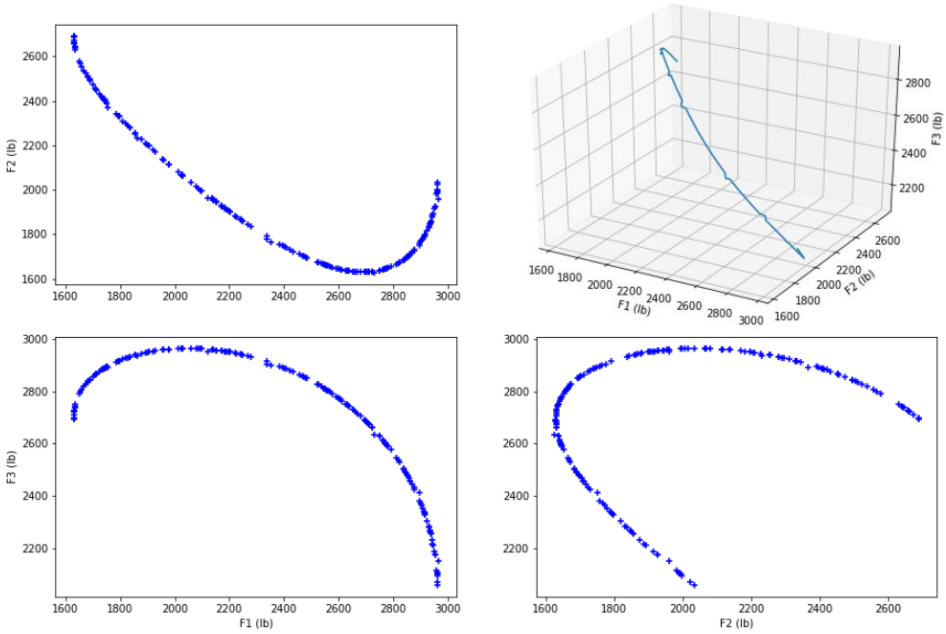

Figura 8. Frente de Pareto, iteración 5.

Como se observa en la figura 8, al ejecutar el algoritmo con 60 generaciones, se observa que el Frente de Pareto se ve más definido, esto debido a la cantidad de puntos que lo conforman, sin embargo, continúa con algunas irregularidades, por lo que se decide aumentar de nuevo la cantidad de generaciones. Los resultados se muestran en la figura 9.

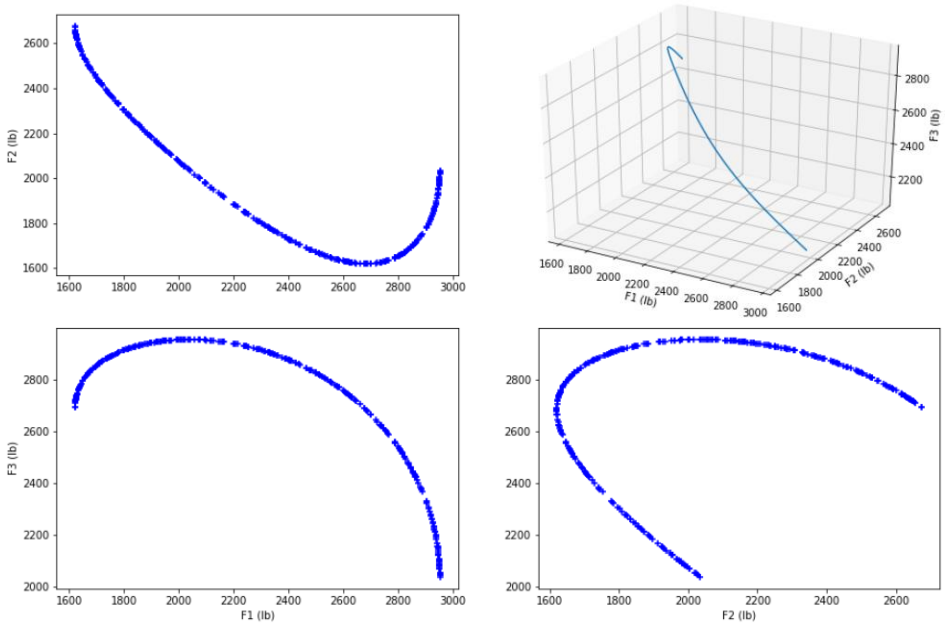

Figura 9. Frente de Pareto, iteración 6.

Como se observa en la figura 9, con una cantidad de 80 generaciones se obtiene un Frente de Pareto bastante abundante y definido. Estas iteraciones fueron realizadas con una población de 100 individuos, por lo cual, ahora se reducirá este valor para encontrar una cantidad óptima.

En la figura 10, se muestran los resultados para 80 generaciones, con una población de 10 individuos.

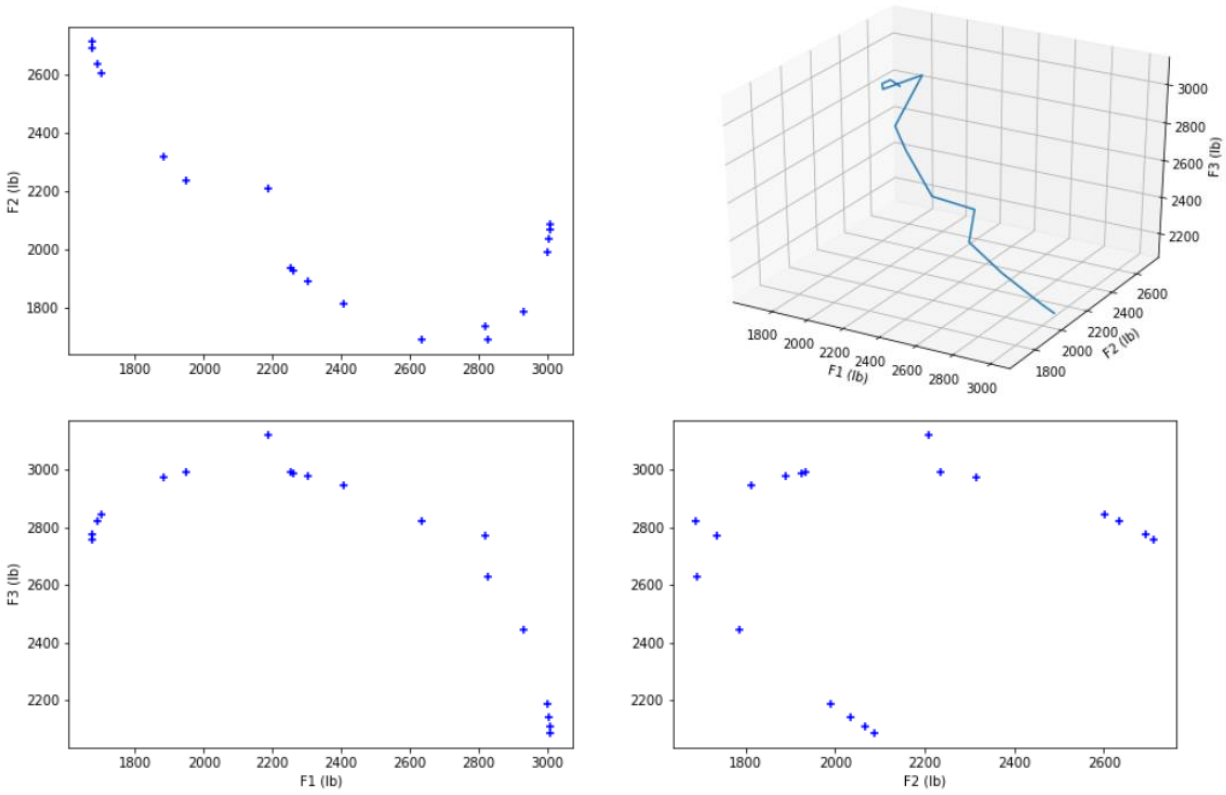

Figura 10. Frente de Pareto, iteración 7.

Como se muestra en la figura 10, 10 individuos por generación no permiten que el algoritmo logre encontrar suficientes puntos del frente de Pareto, por lo cual, se decide continuar con una cantidad mayor.

Debido a lo anterior, en la figura 11, se muestra el frente de Pareto con una población de 25 individuos.

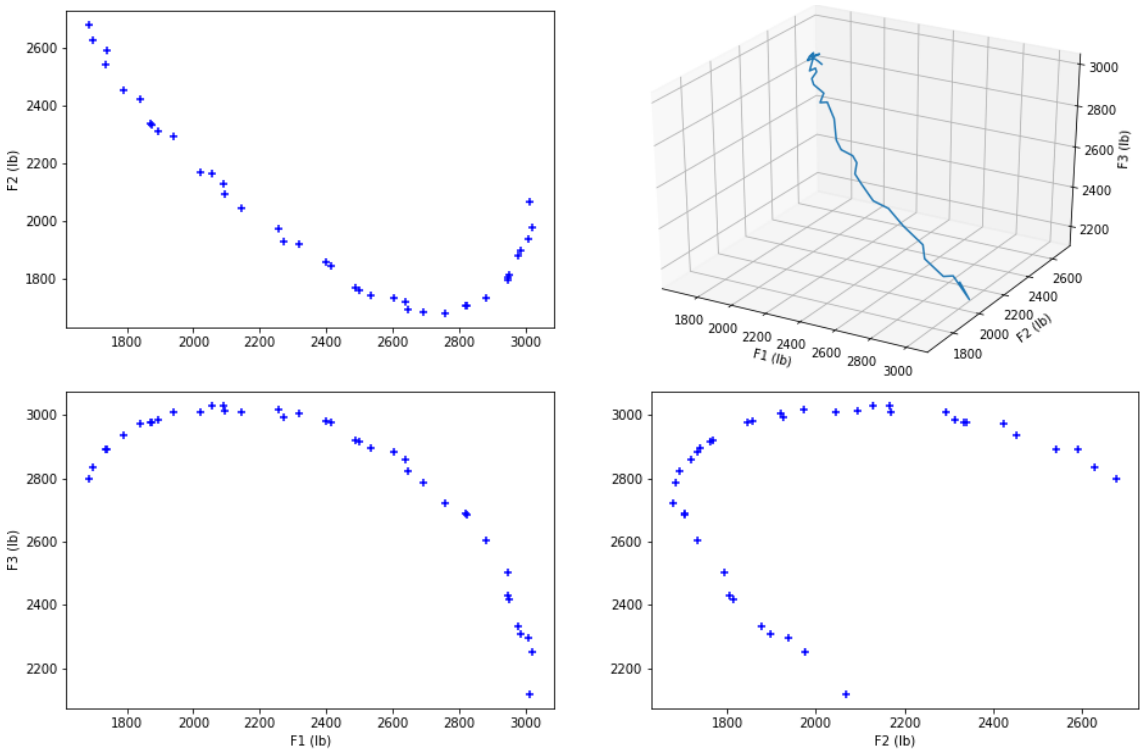

Figura 11. Frente de Pareto, iteración 8.

Como se observa en la figura 11, 25 individuos continúan siendo insuficientes para sacar provecho de la capacidad del algoritmo. Por lo cual, en la figura 12, se muestra el frente de Pareto con 40 individuos por población.

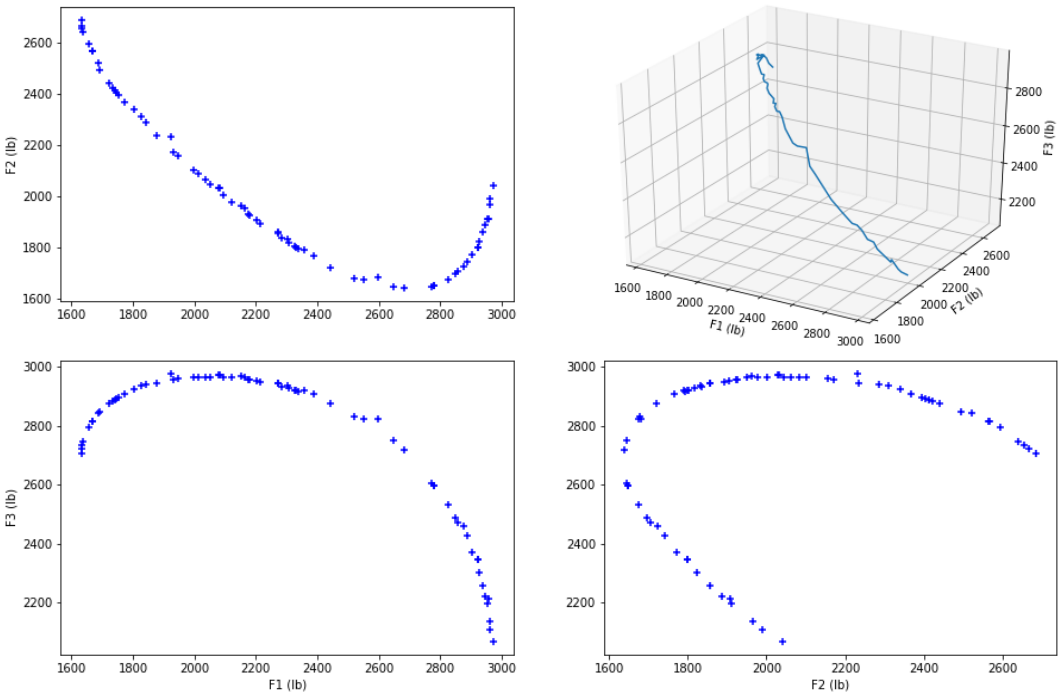

Figura 12. Frente de Pareto, iteración 9.

Al observar la figura 12, se aprecia un Frente de Pareto más robusto y definido, sin embargo, continúa con algunas secciones irregulares. En la figura 13, se muestra la décima iteración, con 55 individuos por población.

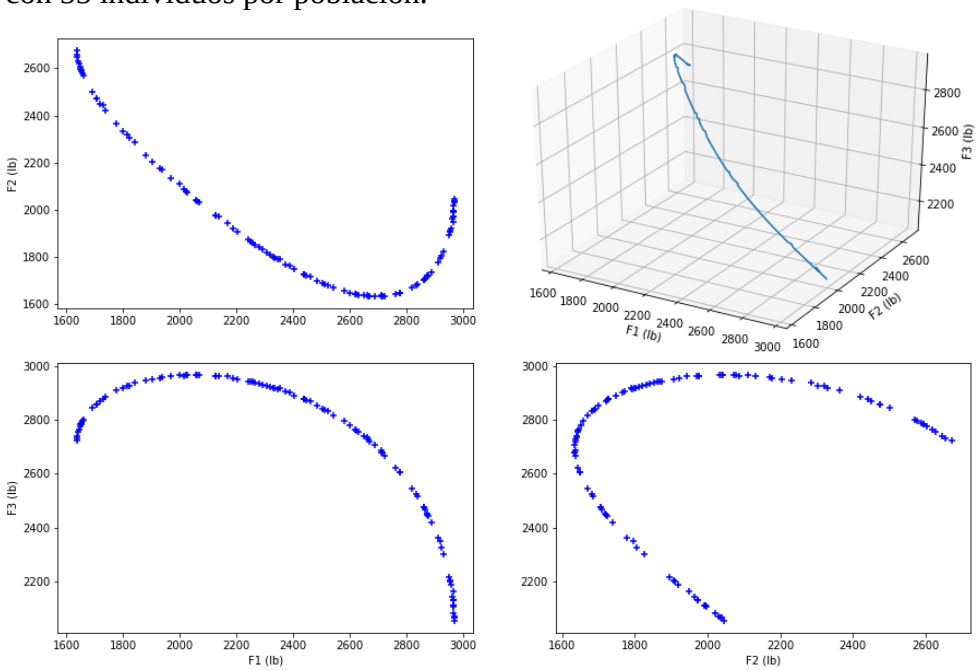

Figura 13. Frente de Pareto, iteración 10.

Al observar detalladamente la figura 13, se nota que, a pesar de obtener una curva suave, aún existen pequeñas irregularidades en el frente de Pareto, motivo por el cual, se decide incrementar una vez más la cantidad de individuos de la población a 70. Dichos resultados se observan en la figura 14.

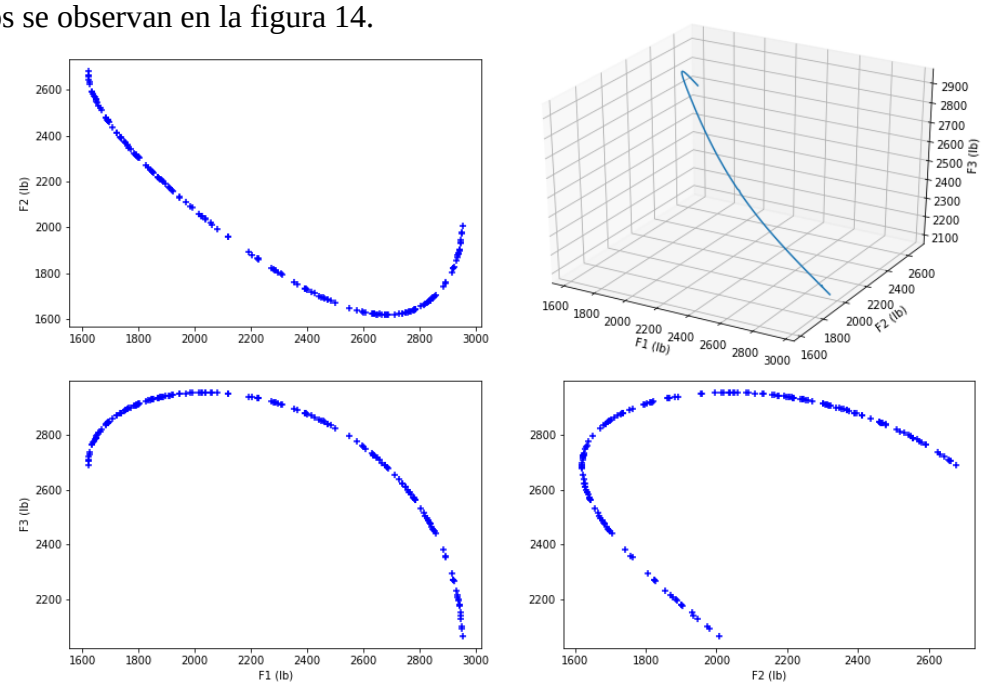

Figura 14. Frente de Pareto, iteración 11.

Finalmente, en la figura 14, se observa un frente de Pareto muy bien definido. Esto se debe a que para 80 generaciones de 70 individuos cada una, el algoritmo fue capaz de encontrar 183 puntos que se encuentran en dicho frente. Por lo anterior, se considera que los valores anteriormente mencionados son los óptimos para la cantidad de generaciones y el tamaño de la población. Asimismo, es importante mencionar que el hiperparámetro de la cantidad de hijos, está directamente ligado al tamaño de la población para los efectos de la solución de este problema, ya que se consideró siempre que dicha cantidad sea siempre la mita del tamaño de la población.

Ahora bien, se va a continuar el estudio de los hiperparámetros que se encuentran ligados principalmente a la variabilidad, lo cuales son la probabilidad de cruzamiento y de mutación. Para esto, se va a utilizar principalmente la gráfica de la desviación estándar frente a la cantidad de generaciones, para observar si existen una tendencia a mayor variabilidad o presión selectiva.

Comenzando con la probabilidad de mutación, en la figura 15 se muestra la desviación estándar para una probabilidad de 10%.

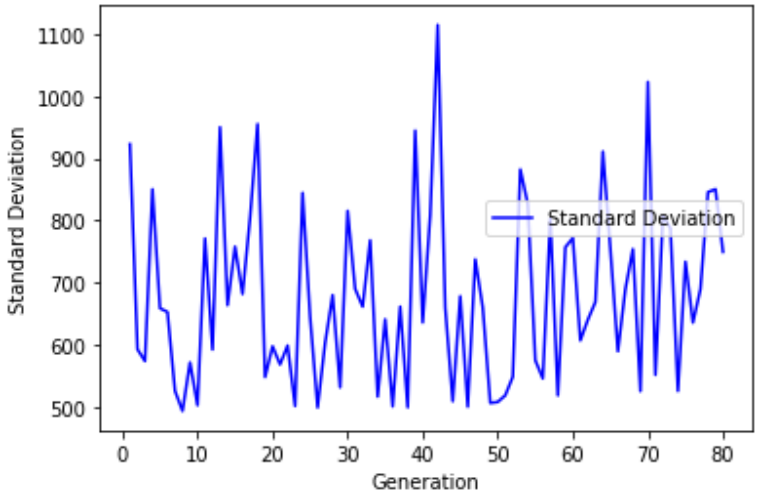

Figura 15. Desviación estándar respecto a las generaciones, iteración 12.

Como se observa en la figura 15, para una probabilidad de mutación del 10%, se encuentra un escenario bastante inclinado a la variabilidad, ya que se muestra una gráfica bastante inestable.

A modo de prueba, en la figura 16, se muestra la desviación estándar de cada generación para un caso donde existe 0% de probabilidad de que se dé la mutación, esto con el fin de determinar que tanto influye dicho hiperparámetro en el comportamiento del algoritmo evolutivo.

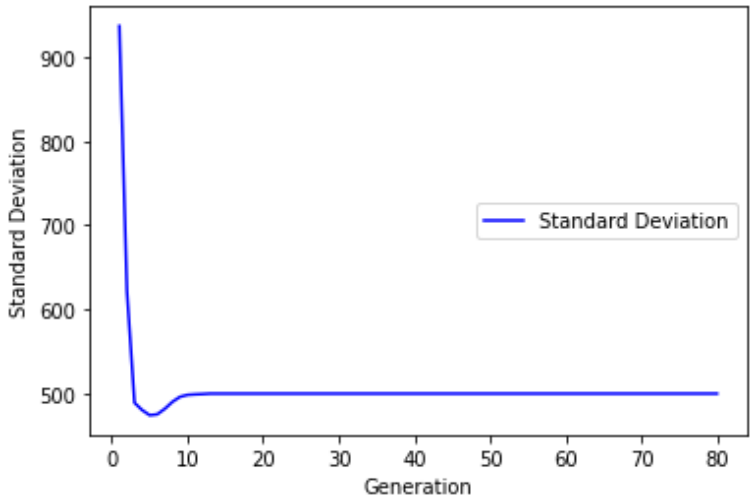

Figura 16. Desviación estándar respecto a las generaciones, iteración 13.

Como se muestra en la figura 16, efectivamente reducir eliminar la probabilidad de mutación produjo que el comportamiento del algoritmo se inclinara bastante a la presión selectiva, ya que se observa una tendencia en la gráfica que se llega a estabilizar. Por lo cual se puede decir que, para el caso de este algoritmo, la mutación juega un papel bastante importante en el comportamiento de este, es decir, define la tendencia entre variabilidad, o presión selectiva.

Sin embargo, se observa en la figura 16, el frente de Pareto para esta misma iteración anterior, y se observa un frente bastante deficiente, ya que está conformado por muy pocos puntos. Esto se debe a que la mutación en el algoritmo es un operador bastante importante, ya que permite la exploración de zonas en el espacio de alelos que el algoritmo por sí solo no alcanza a conocer.

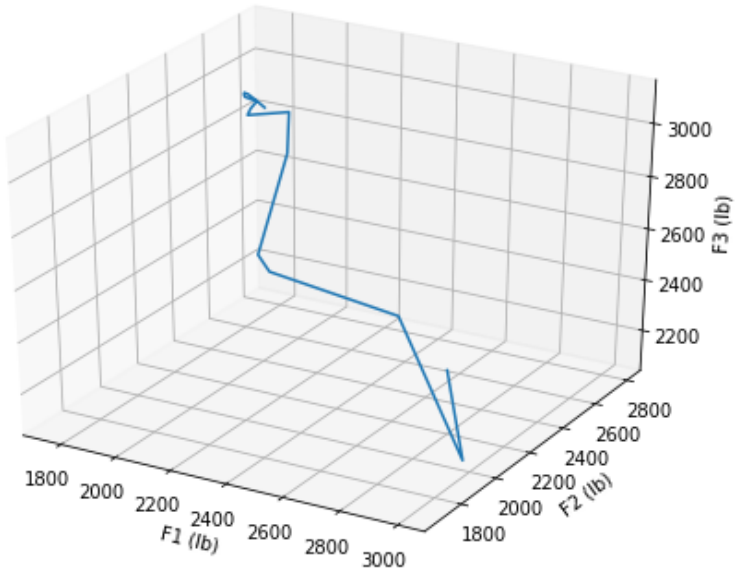

Figura 17. Frente de Pareto, iteración 13

Debido al comportamiento observado anteriormente, es notable que se debe incrementar la probabilidad de mutación hasta un punto óptimo donde permita obtener un frente de Pareto definido, pero que no llegue a favorecer la variabilidad hasta un punto donde el algoritmo sea meramente aleatorio. Por este motivo, en la figura 18, se muestra la desviación estándar en cada generación para una iteración con 2% de probabilidad de mutación. Además, en la figura 19, se muestra el frente de Pareto asociado a esta iteración.

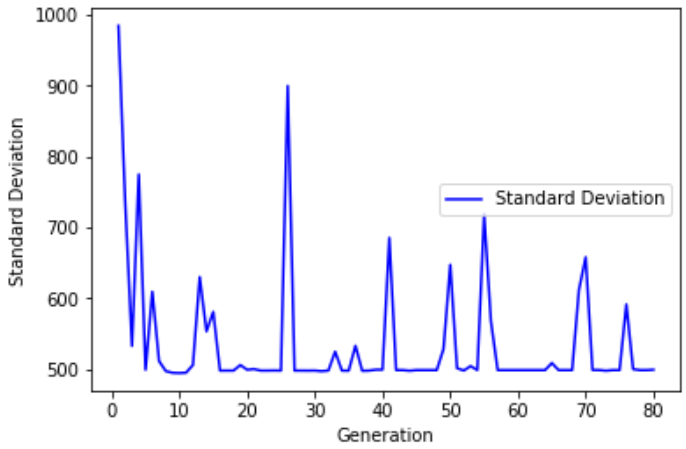

Figura 18. Desviación estándar respecto a las generaciones, iteración 14.

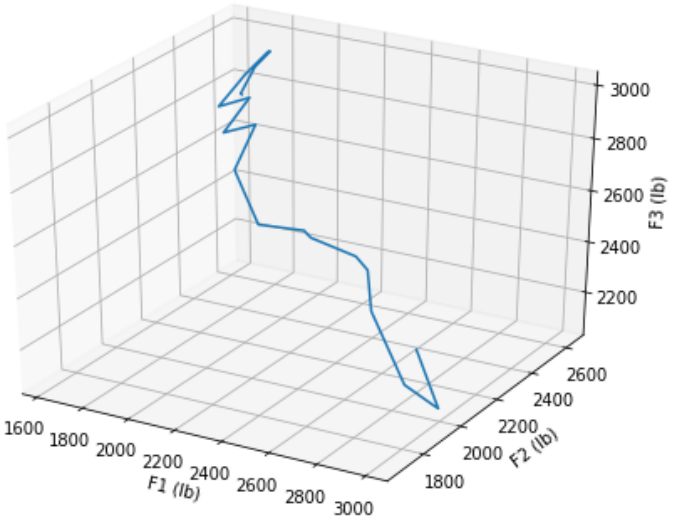

Figura 19. Frente de Pareto, iteración 14

Al analizar la figura 18, se observa una desviación estándar que tiende a estabilizarse, pero, aun así, se inestabiliza por la aparición de picos abruptos, lo cual evidencia que el modelo empieza a tender cada vez más a dejar influir más por la variabilidad y no tanto por la presión selectiva. Sin embargo, al observar la figura 19, la cual corresponde al frente de Pareto para dicha iteración, se puede notar una necesidad de incluir más variabilidad en el algoritmo, para que este sea capaz de encontrar más puntos en el frente.

Debido a lo anterior, en la figura 20, se muestra la desviación estándar para cada generación, con un 5% de probabilidad de mutación, esto con la finalidad de favorecer más la variabilidad de las soluciones que propone el algoritmo.

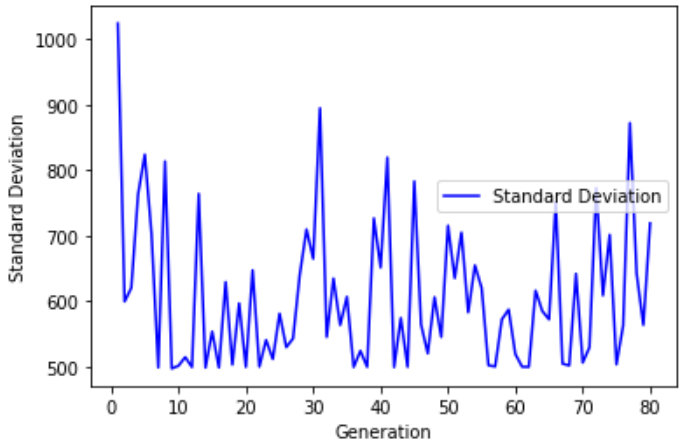

Figura 20. Desviación estándar respecto a las generaciones, iteración 15.

En la figura 20 se observa un comportamiento mucho más dado a la variabilidad, el cuál se considera óptimo para el funcionamiento específico de este algoritmo, ya que para esta iteración se logró obtener un frente de Pareto bastante suave y definido, bastante similar al que se observa en la figura 14. Por lo anterior, se define un 5% como el valor óptimo para la probabilidad de mutación.

Por otra parte, en cuanto al cruzamiento, todas las iteraciones anteriores se han realizado con un 80% probabilidad de que se aplique este operador. Por lo cual, en la figura 21, se muestra el gráfico de desviación estándar obtenido con una probabilidad de cruzamiento del 50%, esto con el fin de analizar si la variabilidad se ve afectada.

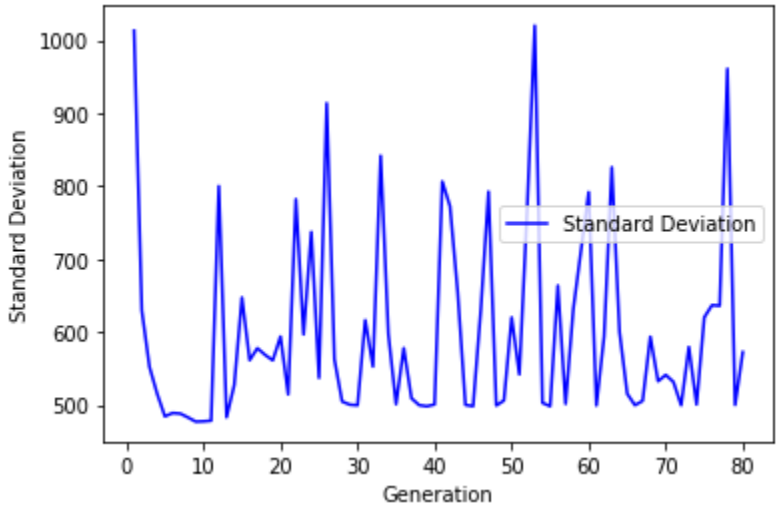

Figura 21. Desviación estándar respecto a las generaciones, iteración 16.

Como se observa en la figura 21, la reducción en la probabilidad de cruzamiento prácticamente no mostró un cambio en la variabilidad del modelo, por lo que se va a continuar reduciendo dicha probabilidad, en este caso a un 40%, para poder analizar resultados en condiciones más extremas.

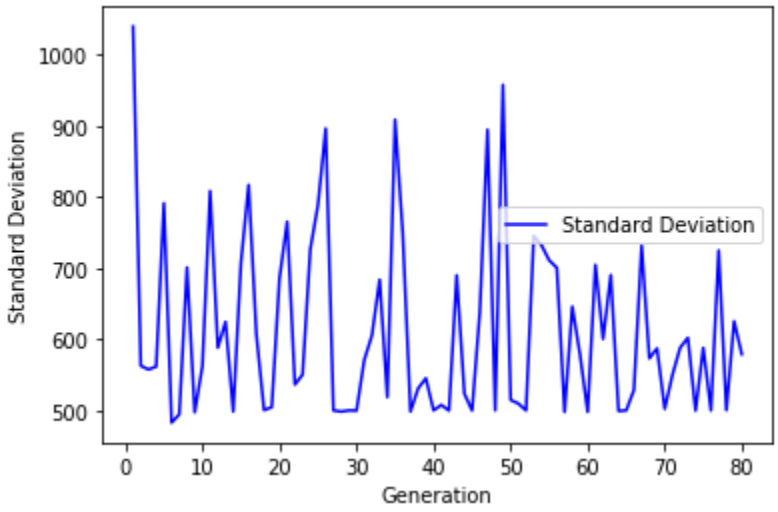

Figura 22. Desviación estándar respecto a las generaciones, iteración 17.

De acuerdo con la figura 22, reducir la probabilidad de cruzamiento a un 40% provoca que el modelo tome en cuenta un poco menos la variabilidad y quiera tender a estabilizar su desviación estándar para las últimas generaciones, ya que se observa que al final del gráfico, los picos obtenidos no son tan abruptos como lo son en la figura 21, para el caso del 50%. A continuación, se va a continuar disminuyendo la probabilidad de cruzamiento, específicamente a un 30%, para observar si este comportamiento incrementa conforme se disminuye dicha probabilidad.

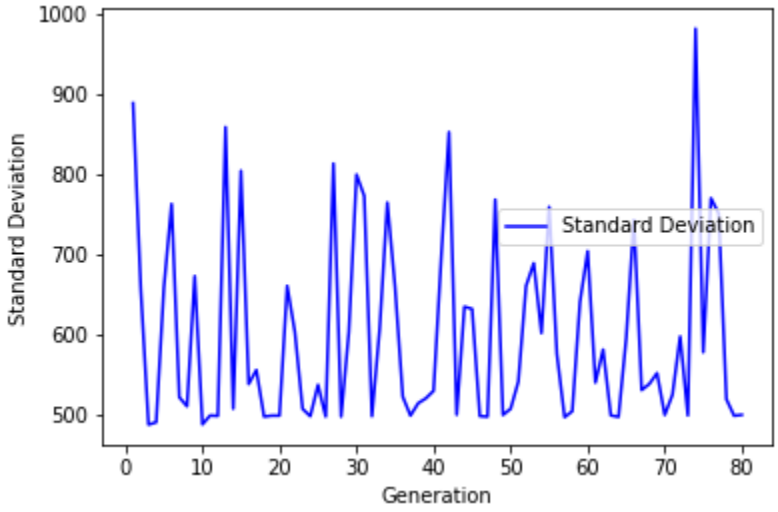

Figura 23. Desviación estándar respecto a las generaciones, iteración 18.

Como se puede observar en la figura 23, un 30% de probabilidad de cruzamiento provoca, contrario a lo esperado, un incremento en la variabilidad, ya que se muestran picos mucho más abruptos para las generaciones finales. Por lo anterior, debido a que se evidencia que el modelo tiende principalmente a la variabilidad, se decide definir un 40% como la probabilidad de cruzamiento, esto para tratar de disminuir un poco la variabilidad en comparación con las alternativas obtenidas con otros porcentajes.

Es importante mencionar que, para ninguna de las iteraciones realizadas para la calibración de la probabilidad de cruzamiento, el frente de Pareto se vio afectado, por este motivo fue que el análisis anterior se basó únicamente en el comportamiento de la desviación estándar, la cual tampoco varió gran medida.

Finalmente, el último hiperparámetro a analizar, es la cantidad de individuos utilizados en cada torneo. Para todas las iteraciones anteriores, se utilizó el parámetro por defecto de 2 individuos, sin embargo, para experimentar más con el comportamiento del algoritmo, en las figuras 24, y 25, se muestran los casos donde se utilizan 3 y 5 individuos respectivamente para cada torneo.

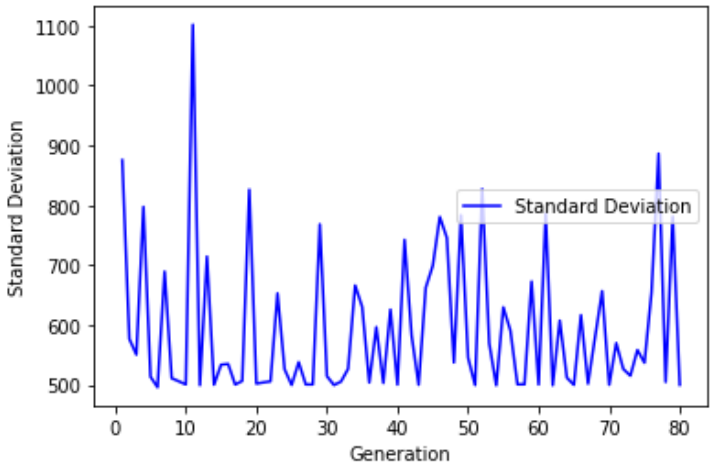

Figura 24. Desviación estándar respecto a las generaciones, iteración 19.

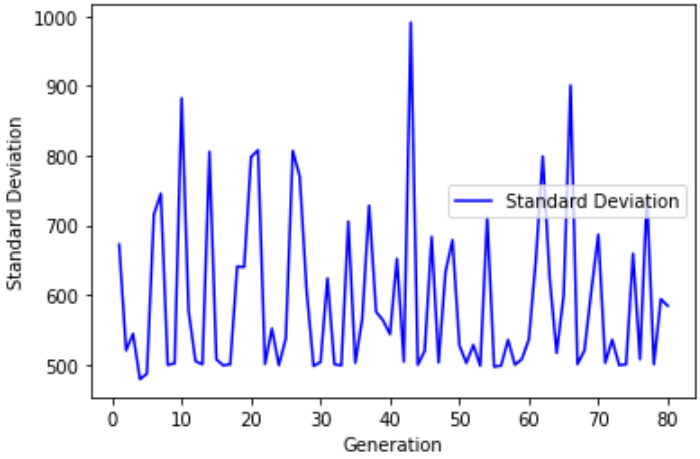

Figura 25. Desviación estándar respecto a las generaciones, iteración 20.

Como se observa en las figuras 24 y 25, el aumento de la cantidad de individuos participantes en cada torneo no afecta de manera relevante la desviación estándar en las generaciones del modelo, por l cual tampoco afecta la variabilidad. Por este motivo, se decide continuar con valor definido por defecto de 2 individuos por cada torneo.

Gracias al estudio realizado anteriormente, a continuación, se muestran los hiperparámetros óptimos para el algoritmo:

- Número de generaciones: 80
- Número de individuos en la población: 70
- Probabilidad de cruzamiento: 0.4
- Probabilidad de mutación:0.01
- Número de hijos: 35
- Tamaño del torneo: 2

Resultados

Gracias a los hiperparámetros definidos anteriormente, se obtuvieron los resultados que se muestran en la figuras 26 y 27, donde se observan la desviación estándar, y el frente de Pareto respectivamente.

Asimismo, en el anexo 2, se muestra la tabla completa de todos los individuos que conforman el frente de Pareto, con su respectiva calidad, y desviación estándar para facilitar la selección de individuos específicos de dicho frente.

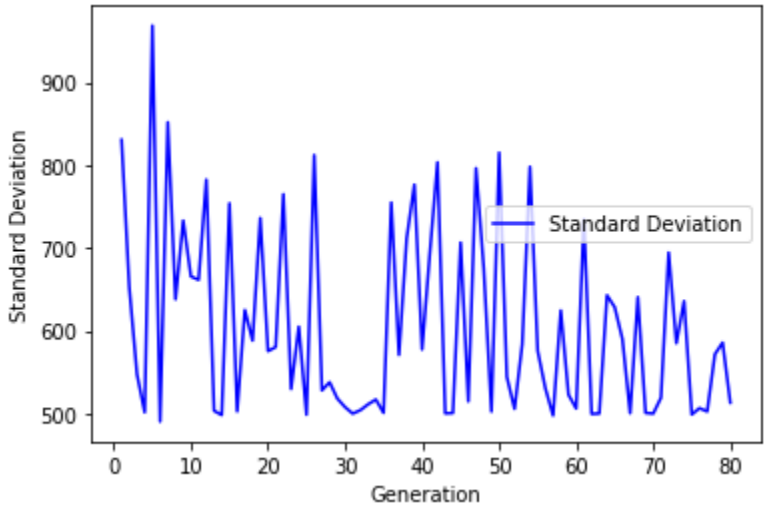

Figura 26. Desviación estándar respecto a las generaciones, resultado final.

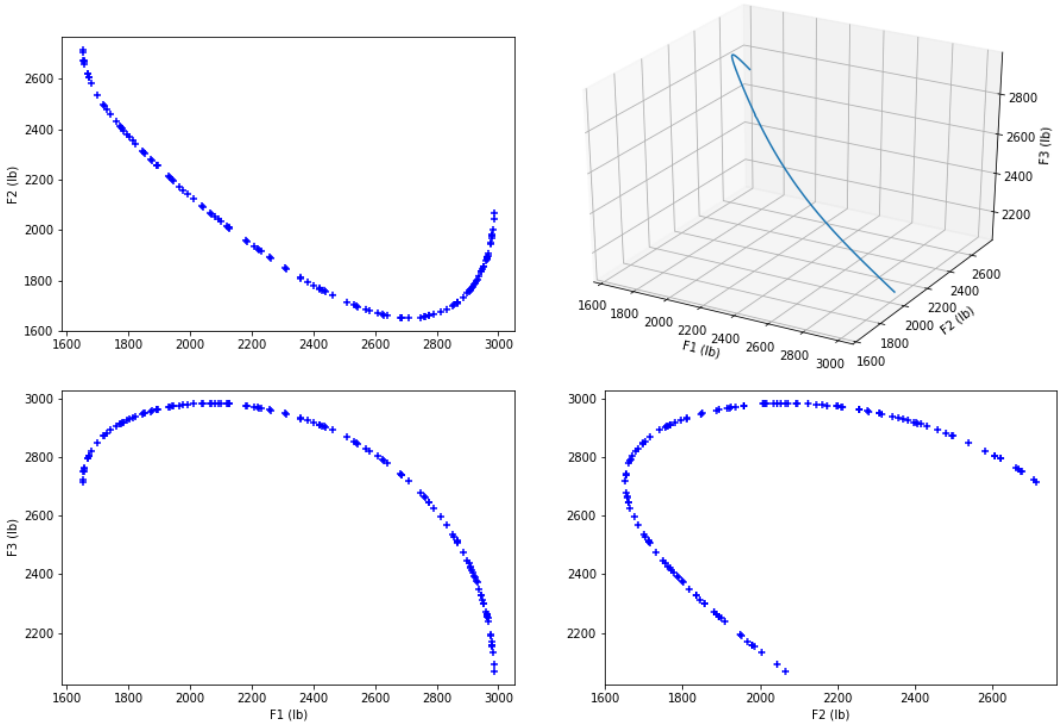

Figura 27. Frente de Pareto, resultado final.

Como se observa en la figura 26, y como se mencionó anteriormente, la desviación estándar evidencia que el modelo tiende mucho más a la variabilidad que a la presión selectiva, sin embargo, para nuestros efectos, esta tendencia permite obtener un frente de Pareto bastante definido, con 133 individuos específicamente (los cuales se muestran en al anexo 2).

Al observar los individuos que pertenecen al frente de Pareto, se puede notar que, pese a la alta variabilidad del modelo, los valores del gen del radio se mantienen constantes en 2.588in. Esto se debe a que las ecuaciones de calidad (o de optimización) tienen dicho gen en el denominador, y como se desea minimizar dichas ecuaciones, el valor óptimo va a tender siempre a acercarse al mayor valor posible.

Ahora bien, para escoger cuáles puntos de este frente se pueden adaptar mejor a la solución del problema planteado, se van a analizar 2 circunstancias, basadas específicamente en la calidad de cada gen. La primera donde se presenta la menor desviación estándar, es decir, donde la fuerza es la más similar entre los 3 pernos. Esta solución es bastante importante, ya que permite que la fuerza se distribuya de la manera más equitativa posible entre los 3 pernos, lo que genera que los 3 tengan esfuerzos similares y, por ende, tiempos de falla también similares. Además, permite el uso de 3 pernos iguales lo que puede abaratar los costos. A continuación, en la tabla 1, se muestra el gen específico del caso planteado anteriormente, donde se busca una equidad en la calidad.

Tabla 1 Gen con menor desviación estándar.

Ángulo (°)	**Radio (in)**	**F1 (lb)**	**F2 (lb)**	**F3 (lb)**	**Desv. Est.**
0.137	2.588	2985.46	2066.16	2069.26	432.637

Por otra parte, la otra solución que se considera importante es la solución donde una de las tres fuerzas toma el valor mínimo posible de todos los valores de fuerza obtenidos para cada función de calidad. Esto se debe a que al minimizar una fuerza frente a las otras 2 permite poder utilizar un perno específico para cada caso, donde el perno que se somete a la menor fuerza puede ser mucho más barato en comparación a los 3 pernos iguales planteados en el caso anterior.

En la tabla 2, se muestra el gen específico, que permite obtener esta solución.

Tabla 2. Gen con menor desviación estándar.

Ángulo (°)	Radio (in)	F1 (lb)	F2 (lb)	F3 (lb)	Desv. Est.
179.940	2.588	1652.13	2713.73	2714.76	500.686

Sin embargo, una fuerza menor frente a 2 fuerzas mucho mayores provoca que se necesiten pernos mucho más robustos para estos 2 otros casos, por lo que finalmente, para tomar la mejor decisión, basada en el factor económico, se debería primero realizar un análisis de resistencia para encontrar los pernos que logren soportar las fuerzas obtenidas, y luego de esto, cotizar el costo económico de dichos pernos.

Conclusiones

A modo de conclusión, gracias al análisis anteriormente desarrollado, y los resultados obtenidos, se demostró que, para este algoritmo evolutivo, la variabilidad tiende a estar más ligada a la operación de la mutación que al cruzamiento, ya que cuando se redujo la probabilidad de mutación, se disminuyó la variabilidad de las soluciones, lo cual afectó gravemente el frente de Pareto.

De manera similar, se demostró que el operador de cruzamiento no genera cambios relevantes en la variabilidad o presión selectiva de la red, ya que, a pesar de iterar con diferentes probabilidades para esta operación, los resultados no se vieron afectados de una manera significativa.

Por otra parte, los individuos que fueron seleccionados como soluciones óptimas para el problema planteado se escogieron gracias a la desviación estándar que tienen asociada a la calidad de los objetivos. Para un primer caso se buscó al individuo con la menor desviación estándar, mientras que, para el otro caso, se buscó al individuo con la mayor desviación, para proponer 2 posibles soluciones. Sin embargo, es importante mencionar que, en este caso, para determinar la solución óptima desde la perspectiva económica, se debe realizar un análisis más profundo con otros factores que no se consideraron en el presente documento.

Por lo anterior, se recomienda que, para continuar con el desarrollo del problema, y se pueda obtener una solución única desde el aspecto monetario, se implemente un algoritmo evolutivo que pueda minimizar el costo de los pernos, tomando en cuenta aspectos como los

esfuerzos debidos a las fuerzas, la resistencia del material elegido para los pernos, la disponibilidad en el mercado, y su precio.

De igual manera, es importante mencionar que el estudio de los hiperparámetros del algoritmo se hizo con base en las gráficas de desviación estándar y del frente de Pareto. Esto para tomar en cuenta tanto una medición relativa de la influencia de la variabilidad y la presión selectiva en el modelo, como las posibles soluciones al problema y la estabilidad que estas presentaban.

Finalmente, cabe destacar que el incremento de las generaciones y de la cantidad de individuos en las poblaciones permite encontrar más puntos en el frente de Pareto, sin embargo, este criterio no es el único que controla la cantidad de puntos encontrados, ya que se demostró que la mutación (como operador de variabilidad) tiene un papel bastante importante en el descubrimiento de los puntos del frente de Pareto.

Referencias

DEAP documentation. DEAP documentation - DEAP 1.3.3 documentation. (2022, August 8). Obtained from: https://deap.readthedocs.io/en/master/index.html

DEAP documentation. DEAP documentation - DEAP 2.0.0.a0 documentation. (2021, February 13). Obtained from: https://deap.readthedocs.io/en/devel/index.html

Anexos

Anexo 1: Codificación

```
# librerías

import random
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

!pip install deap

from deap import base
from deap import creator
from deap import tools

# Fixed Parameters

F = 2000 # Fuerza total en lb
NDIM = 2 # Number of dimensions of the individual

LOW1, UP1 = 0, 180 # Bounds on the first gene (ángulo alfa)
LOW2, UP2 = (4/9)*math.sqrt(3), 21/8 # Bounds on the second gene (radi
o)
BOUNDS = [(LOW1, UP1)] + [(LOW2, UP2)]

# Hyperparameters

NGEN = 80 # Number of Generation
MU = 70 # Number of individuals in population
CXPB = 0.4 # Crossover probability
MUTPB = 0.05 #0.05 # Mutation probability
NH = 35 # Number of children
TS = 2 # Tournament Size



def init_opti():

  toolbox = base.Toolbox()
  creator.create("FitnessMulti", base.Fitness, weights=(-1.0,-1.0,-1.0))
  creator.create("Individual", list, fitness=creator.FitnessMulti)
```

```
  # registro de operadores
  toolbox.register("individual", gen_tuple, icls = creator.Individual,
                   ranges=BOUNDS)
  toolbox.register("population", tools.initRepeat, list,
                   toolbox.individual)
  toolbox.register("evaluate", evaluation)
  toolbox.register("cross", tools.cxUniform, indpb = CXPB)
  toolbox.register("mutate", mutar_ind, prob = MUTPB,
                   icls = creator.Individual, ranges=BOUNDS)
  toolbox.register("select", tools.selTournament, tournsize = TS)
  toolbox.register("delete", tools.selWorst)

# función para generar los genes
def gen_tuple(icls, ranges):
    genome = list()
    for p in BOUNDS:
        genome.append(np.random.uniform(*p))
    return icls(genome)

#función de calidad
def evaluation(ind):
  a_grados = ind[0] # ángulo en grados
  a = (a_grados*math.pi)/180 # ángulo en radianes
  r = ind[1]

  objective1 = (F/3)*math.sqrt(pow((9/r)*math.sin(a), 2)
                + pow(1 + (9/r)*math.cos(a), 2))
  objective2 = (F/3)*math.sqrt(pow((9/r)*math.sin(a + (2/3)*math.pi), 2)
                + pow(1 + (9/r)*math.cos(a + (2/3)*math.pi), 2))
  objective3 = (F/3)*math.sqrt(pow((9/r)*math.sin(a + (4/3)*math.pi), 2)
                + pow(1 + (9/r)*math.cos(a + (4/3)*math.pi), 2))
  return objective1, objective2, objective3

#función de mutación
def mutar_ind(ind, prob, icls, ranges):

  # mutación del 1er gen
  n = random.random()
  if n < prob:
    ind[0] = gen_tuple(icls, ranges)[0]

  # mutación del 2do gen
  n = random.random()
  if n < prob:
    ind[1] = gen_tuple(icls, ranges)[1]
```

```
  return ind

def main():

  #statistics
  stats = tools.Statistics(key=lambda ind: ind.fitness.values)
  stats.register("std", np.std)
  pareto = tools.ParetoFront()
  logbook = tools.Logbook()

  pop = toolbox.population(n=MU)

  print("Start of evolution")

  # Evaluate the entire population
  fitnesses = list(map(toolbox.evaluate, pop))
  for ind, fit in zip(pop, fitnesses):
    ind.fitness.values = fit

  print("  Evaluated %i individuals" % len(pop))

  # Extracting all the fitnesses of de individuals
  fits = [ind.fitness.values[0] for ind in pop]

  # Variable keeping track of the number of generations
  g = 0

  # Begin the evolution
  while g < NGEN:

    g = g + 1

    # Select the next generation individuals
    offspring = toolbox.select(pop, NH)

    # Clone the selected individuals
    offspring = [toolbox.clone(ind) for ind in offspring]

    # Apply crossover and mutation on the offspring

    for ind1, ind2 in zip(offspring[0::2], offspring[1::2]):
        # cross two individuals
        toolbox.cross(ind1, ind2)
```

```
    for mutant in offspring:
      # mutate an individual
      toolbox.mutate(mutant)
      del mutant.fitness.values

    # Evaluate the individuals with an invalid fitness
    fitnesses = map(toolbox.evaluate, offspring)
    for ind, fit in zip(offspring, fitnesses):
      ind.fitness.values = fit

    # Eliminar individuos
    for i in range(NH):
      selected = toolbox.delete(pop, 1)[0]
      pop.remove(selected)

    # Añadir nuevos a población
    pop.extend(offspring)

    pareto.update(pop)
    record = stats.compile(pop)
    logbook.record(gen=g, **record)

  print("-- End of (successful) evolution --")

  #guardar estadisticas
  gen, std = logbook.select("gen", "std")

  return pareto, gen, std

# grafico del pareto
def plot(data):

  # Se obtienen los datos
  x, y, z = zip(*[ind.fitness.values for ind in data])
  x = np.array([x])
  y = np.array([y])
  z = np.array([z])

  fig = plt.figure()
  fig.set_size_inches(15,10)

  # Pareto F2 vs F1
  axe = plt.subplot2grid((2,2),(0,0))
  axe.set_ylabel('F2 (lb)')
  axe.scatter(x, y, c='b', marker='+')
```

```
  # Pareto F3 vs F1
  axe = plt.subplot2grid((2,2),(1,0))
  axe.set_ylabel('F3 (lb)')
  axe.set_xlabel('F1 (lb)')
  axe.scatter(x, z, c='b', marker='+')

  #Pareto F3 vs F2
  axe = plt.subplot2grid((2,2),(1,1))
  axe.set_xlabel('F2 (lb)')
  scat = axe.scatter(y, z, c='b', marker='+')

  plt.show()

  # Pareto en 3D
  fig = plt.figure()
  fig.set_size_inches(8,6)

  axe = fig.add_subplot(111,projection='3d')
  axe.set_xlabel('F1 (lb)')
  axe.set_ylabel('F2 (lb)')
  axe.set_zlabel('F3 (lb)')

  axe.plot_wireframe(x, y, z)
  plt.show()

#grafico de la desviación estándar
def plot_stats(gen, std):
  fig, ax1 = plt.subplots()
  line = ax1.plot(gen, std, "b-", label="Standard Deviation")
  ax1.set_xlabel("Generation")
  ax1.set_ylabel("Standard Deviation")

  labs = [l.get_label() for l in line]
  ax1.legend(line, labs, loc="center right")

  plt.show()

# tabulación del pareto
def tabular(pareto):

  # obtención de los datos
  F1, F2, F3 = zip(*[ind.fitness.values for ind in pareto])
```

```
  # listas para almacenar el contenido de la tabla
  tabla = list()
  std = list()

  # redondeo y ordenamiento de los datos
  for number in range(len(pareto)):
    fila = list()
    fila.append(round(pareto[number][0], 3))
    fila.append(round(pareto[number][1], 3))
    fila.append(round(F1[number], 2))
    fila.append(round(F2[number], 2))
    fila.append(round(F3[number], 2))
    desv_est = [F1[number], F2[number], F3[number]]
    fila.append(round(np.std(desv_est), 3))
    std.append(np.std(desv_est))
    tabla.append(fila)

  # etiquetas de la tabla
  labels = ["Ángulo (°)", "Radio (in)", "F1 (lb)", "F2 (lb)",
            "F3 (lb)", "Std"]

  # confección de la tabla
  df = pd.DataFrame(tabla, columns = labels)
  pd.set_option('display.max_rows', None)
  print(df)

  # selección del individuo con menor desviación estándar
  min_std = round(min(std), 3)
  print("El individuo con la menor desviación estándar es:")
  select = list()
  for element in tabla:
    if min_std == element[5]:
      select.append(element)
  df = pd.DataFrame(select, columns = labels)
  print(df)


# ejecución del programa
init_opti()
pareto, gen, std = main()
plot_stats(gen, std)
plot(pareto)
tabular(pareto)
```

Anexo 2: Tabulación del frente de Pareto

Tabla 3. Frente de Pareto

	Ángulo (°)	Radio (in)	F1 (lb)	F2 (lb)	F3 (lb)	Desv. Est.
0	179.940	2.588	1652.13	2713.73	2714.76	500.686
1	179.030	2.588	1652.27	2705.85	2722.54	500.644
2	175.572	2.588	1654.92	2675.03	2751.23	499.812
3	175.309	2.588	1655.26	2672.63	2753.35	499.706
4	174.622	2.588	1656.25	2666.34	2758.86	499.400
5	173.891	2.588	1657.44	2659.59	2764.65	499.030
6	169.769	2.588	1666.94	2620.50	2796.09	496.104
7	169.709	2.588	1667.12	2619.91	2796.53	496.051
8	168.279	2.588	1671.53	2605.95	2806.93	494.712
9	168.224	2.588	1671.71	2605.41	2807.32	494.657
10	165.840	2.588	1680.32	2581.70	2824.04	492.080
11	161.613	2.588	1699.23	2538.44	2851.80	486.587
12	157.959	2.588	1719.15	2499.88	2873.84	481.048
13	157.555	2.588	1721.56	2495.54	2876.16	480.398
14	156.197	2.588	1729.89	2480.93	2883.79	478.174
15	154.141	2.588	1743.31	2458.56	2894.85	474.693
16	151.692	2.588	1760.48	2431.57	2907.23	470.420
17	150.204	2.588	1771.51	2415.02	2914.32	467.786
18	149.453	2.588	1777.26	2406.61	2917.78	466.449
19	149.200	2.588	1779.21	2403.77	2918.92	466.000
20	148.553	2.588	1784.28	2396.49	2921.81	464.847
21	147.299	2.588	1794.32	2382.35	2927.23	462.620
22	146.137	2.588	1803.88	2369.17	2932.04	460.569
23	144.991	2.588	1813.55	2356.13	2936.59	458.566
24	143.724	2.588	1824.49	2341.65	2941.38	456.383
25	141.302	2.588	1846.17	2313.81	2949.88	452.333
26	140.772	2.588	1851.03	2307.69	2951.62	451.475
27	140.676	2.588	1851.91	2306.60	2951.93	451.322
28	138.487	2.588	1872.47	2281.27	2958.62	447.914
29	138.030	2.588	1876.86	2275.97	2959.93	447.231
30	136.345	2.588	1893.24	2256.42	2964.46	444.815
31	136.274	2.588	1893.95	2255.58	2964.65	444.716
32	136.243	2.588	1894.25	2255.22	2964.73	444.673
33	132.629	2.588	1930.65	2213.19	2972.91	440.116
34	132.287	2.588	1934.17	2209.21	2973.58	439.733
35	131.484	2.588	1942.49	2199.87	2975.08	438.867
36	131.201	2.588	1945.44	2196.58	2975.59	438.573
37	129.136	2.588	1967.20	2172.59	2978.89	436.630

38	128.091	2.588	1978.37	2160.47	2980.31	435.784
39	126.667	2.588	1993.74	2143.99	2981.96	434.785
40	124.959	2.588	2012.40	2124.30	2983.53	433.832
41	122.440	2.588	2040.29	2095.43	2985.00	432.927
42	122.138	2.588	2043.67	2091.98	2985.11	432.860
43	120.254	2.588	2064.84	2070.57	2985.46	432.639
44	119.714	2.588	2070.94	2064.48	2985.46	432.640
45	118.747	2.588	2081.90	2053.59	2985.34	432.713
46	117.816	2.588	2092.51	2043.16	2985.09	432.869
47	117.040	2.588	2101.37	2034.50	2984.77	433.064
48	115.356	2.588	2120.67	2015.87	2983.77	433.685
49	115.140	2.588	2123.16	2013.49	2983.60	433.784
50	114.806	2.588	2127.01	2009.81	2983.34	433.947
51	114.693	2.588	2128.30	2008.58	2983.25	434.004
52	110.127	2.588	2181.14	1959.39	2977.79	437.283
53	109.903	2.588	2183.75	1957.02	2977.44	437.491
54	107.829	2.588	2207.87	1935.37	2973.80	439.605
55	106.839	2.588	2219.38	1925.20	2971.84	440.730
56	106.398	2.588	2224.51	1920.70	2970.91	441.255
57	105.826	2.588	2231.16	1914.90	2969.66	441.955
58	103.579	2.588	2257.29	1892.50	2964.27	444.920
59	103.011	2.588	2263.89	1886.94	2962.78	445.719
60	99.137	2.588	2308.75	1850.19	2951.32	451.622
61	98.725	2.588	2313.50	1846.41	2949.97	452.290
62	94.823	2.588	2358.25	1811.96	2935.86	458.890
63	94.689	2.589	2358.64	1809.68	2934.17	459.114
64	92.740	2.588	2381.90	1794.64	2927.39	462.551
65	90.994	2.588	2401.59	1780.72	2919.80	465.654
66	89.404	2.588	2419.39	1768.56	2912.48	468.481
67	89.346	2.588	2420.04	1768.13	2912.21	468.584
68	88.587	2.588	2428.48	1762.51	2908.58	469.928
69	88.086	2.588	2434.04	1758.87	2906.14	470.812
70	87.475	2.588	2440.79	1754.50	2903.12	471.885
71	85.631	2.588	2461.06	1741.77	2893.65	475.085
72	81.238	2.588	2508.44	1714.50	2869.16	482.318
73	79.158	2.588	2530.38	1703.16	2856.61	485.472
74	78.183	2.588	2540.55	1698.21	2850.52	486.876
75	77.717	2.588	2545.40	1695.93	2847.56	487.529
76	75.359	2.588	2569.58	1685.22	2832.16	490.636
77	74.104	2.588	2582.26	1680.10	2823.65	492.145
78	71.492	2.588	2608.20	1670.78	2805.28	494.937

79	69.871	2.588	2623.98	1665.93	2793.43	496.414
80	69.227	2.588	2630.18	1664.19	2788.63	496.944
81	68.209	2.588	2639.89	1661.69	2780.93	497.713
82	63.679	2.588	2681.82	1654.06	2745.13	500.082
83	63.083	2.588	2687.18	1653.49	2740.24	500.262
84	63.083	2.588	2687.18	1653.49	2740.24	500.262
85	60.664	2.588	2708.51	1652.20	2719.94	500.666
86	56.144	2.588	2746.58	1654.25	2680.22	500.023
87	54.501	2.588	2759.82	1656.43	2665.23	499.342
88	54.187	2.588	2762.32	1656.94	2662.33	499.185
89	52.590	2.588	2774.81	1659.93	2647.43	498.257
90	52.534	2.588	2775.24	1660.05	2646.91	498.221
91	50.518	2.588	2790.54	1664.87	2627.72	496.738
92	47.393	2.588	2813.23	1674.54	2597.20	493.805
93	44.589	2.588	2832.51	1685.44	2569.05	490.571
94	41.427	2.588	2852.97	1700.16	2536.50	486.321
95	40.677	2.588	2857.63	1704.02	2528.66	485.230
96	39.301	2.588	2865.96	1711.46	2514.16	483.155
97	39.238	2.588	2866.34	1711.82	2513.49	483.058
98	39.032	2.588	2867.56	1712.97	2511.31	482.739
99	38.751	2.588	2869.22	1714.56	2508.32	482.301
100	35.781	2.588	2886.08	1732.53	2476.43	477.480
101	33.182	2.588	2899.80	1749.88	2448.04	473.033
102	32.323	2.588	2904.12	1755.93	2438.56	471.532
103	31.410	2.588	2908.60	1762.54	2428.44	469.922
104	30.801	2.588	2911.51	1767.03	2421.67	468.845
105	30.260	2.588	2914.06	1771.09	2415.64	467.884
106	29.531	2.588	2917.42	1776.65	2407.49	466.589
107	28.225	2.588	2923.25	1786.88	2392.80	464.264
108	27.932	2.588	2924.52	1789.21	2389.50	463.744
109	26.708	2.588	2929.70	1799.15	2375.65	461.574
110	26.376	2.588	2931.07	1801.90	2371.88	460.989
111	24.430	2.588	2938.74	1818.36	2349.72	457.594
112	22.564	2.588	2945.56	1834.75	2328.34	454.420
113	22.520	2.588	2945.72	1835.16	2327.83	454.345
114	21.320	2.588	2949.82	1846.00	2314.01	452.363
115	20.163	2.588	2953.56	1856.67	2300.67	450.503
116	20.111	2.588	2953.73	1857.15	2300.07	450.421
117	17.569	2.588	2961.21	1881.30	2270.63	446.555
118	17.061	2.588	2962.59	1886.24	2264.72	445.821
119	17.034	2.588	2962.66	1886.50	2264.41	445.783

120	16.891	2.588	2963.04	1887.89	2262.76	445.581
121	16.289	2.588	2964.61	1893.80	2255.76	444.737
122	16.163	2.588	2964.93	1895.04	2254.30	444.564
123	15.740	2.588	2965.99	1899.22	2249.38	443.988
124	14.759	2.588	2968.33	1909.00	2237.98	442.696
125	10.988	2.588	2975.96	1947.67	2194.10	438.357
126	10.706	2.588	2976.44	1950.62	2190.83	438.076
127	8.990	2.588	2979.10	1968.76	2170.89	436.506
128	7.863	2.588	2980.59	1980.82	2157.83	435.612
129	7.857	2.588	2980.60	1980.88	2157.76	435.607
130	7.513	2.588	2981.02	1984.59	2153.78	435.357
131	5.680	2.588	2982.92	2004.50	2132.60	434.201
132	2.136	2.588	2985.11	2043.69	2091.96	432.859
133	0.137	2.588	2985.46	2066.16	2069.26	432.637