# Optimización del calendario de inspecciones de una flota de aviones mediante un Algoritmo Genético

Dayana Danilsa López Santos

Trabajo de grado presentado para optar al título de Ingeniero Mecánico

Asesor Externo
Ph.D. Juan David Ocampo de los Ríos
Saint Mary's University, San Antonio, TX, EEUU

Asesor Interno
Ph.D.(c) Liliana Marcela Bustamante Góez
Universidad de Antioquia

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería Mecánica
Medellín, Antioquia, Colombia
2023

| | | |
|---|---|---|
| **Cita** | | López Santos [1] |
| **Referencia**<br>Estilo IEEE (2020) | [1] | D. López Santos, “Optimización del calendario de inspecciones de una flota de aviones mediante un Algoritmo Genético”, Trabajo de grado profesional, Ingeniería Mecánica, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2023. |



Grupo de Investigación Diseño Mecánico.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

## RESUMEN

Los grandes esfuerzos a los que son sometidos los elementos estructurales principales (*Principal Structural Element,* PSE) de un avión hace inevitable tanto la iniciación como la propagación de grietas en estos. El Grupo de Dirección de Mantenimiento-3 (*Maintenance Steering Group-3,* MSG-3) contiene una metodología y procesos lógicos específicos para llevar a cabo en el mantenimiento de estos, en esta se especifica que estos elementos deben ser sometidos a inspecciones constantemente con el fin de mantener la confiabilidad de la estructura. Este documento presenta el desarrollo de un Algoritmo Genético (AG) enfocado en optimizar un cronograma de mantenimiento de un PSE, en principio para un solo avión y posteriormente para flota. A grandes rasgos, en esta investigación, el AG es integrado a SMART|DT, con el fin de evaluar la viabilidad de un grupo de combinaciones de cronogramas de inspección, donde luego de un ciclo repetitivo el algoritmo converge hacia un conjunto de soluciones óptimas. Los resultados obtenidos a través de esta metodología permitieron establecer un cronograma de mantenimiento ajustado a la progresión de las grietas en la estructura.

***Palabras clave*** — **Algoritmo Genético, Crecimiento de grieta, Smart|DT, Cronograma de inspecciones.**

## ABSTRACT

The stresses to which the main structural elements (PSE) of an aircraft are subjected make both the initiation and the propagation of cracks in them inevitable. The Maintenance Steering Group-3 (MSG-3) contains a methodology and specific logical processes to carry out in the maintenance of these, in this it is specified that these elements must be submitted to constant inspections in order to maintain the reliability of the structure. This document presents the development of a Genetic Algorithm focused on optimizing a PSE maintenance schedule, initially for a single aircraft and later for a fleet. Broadly speaking, in this research, AG is integrated into SMART|DT, in order to evaluate the feasibility of a group of inspection schedule combinations, where after a repetitive cycle the algorithm converges towards a set of optimal solutions. The results obtained through this methodology allowed establishing a maintenance schedule adjusted to the progression of the cracks in the structure.


**Keywords — Genetic Algorithm, Crack Growth, Smart|DT, Inspection Schedule.**

# I. INTRODUCCIÓN

En términos muy generales, la raíz de una falla en la integridad estructural de un avión puede ser atribuida a dos grandes razones; la primera a una sobrecarga en la que se exceda la resistencia mecánica del elemento y la segunda a la acumulación de daños pequeños pero aditivos, esto último se conoce como falla por fatiga[1] y se encuentra entre las modalidades de falla estructural más comunes en la aviación[2][3].

Dado que falla por fatiga surge como resultado del envejecimiento natural al que son expuestos los sistemas estructurales de los aviones diseñados bajo el concepto de "Damage Tolerance", constantemente en la industria aeronáutica se desarrollan planes de mantenimiento que permitan detectar fallas potenciales, antes que se transformen en fallas funcionales y de este modo conservar la confiabilidad de estos sistemas.

Para evaluar el estado de la estructura de los aviones, generalmente en los programas de mantenimiento aeronáutico se examina cada elemente estructural significativo (SSI) mediante inspecciones periódicas, las cuales tiene como principal objetivo detectar daños asociados a fatiga. Sin embargo, a la hora de diseñar estos cronogramas es un desafío determinar intervalos de inspección eficientes[4]. La importancia de determinar los tiempos correctos en los cuales realizar las inspecciones, radica en que cuando estas se realizan con demasiada frecuencia se tiene un mayor costo asociado al mantenimiento, mientras que lo contrario da como resultado una mayor probabilidad de falla[5].

Como solución a lo anteriormente planteado se han realizado varias investigaciones, en las cuales, si bien se han tenido buenos resultados, aplicarlas aún resulta algo costoso, muy complejo o requieren una alta capacidad de procesamiento[6][7][8][9]. A continuación, se presenta una investigación en la cual se plantea optimizar, en primera instancia, un plan de inspecciones para un determinado elemento estructural de un avión y posteriormente el cronograma de inspecciones en este mismo elemento estructural para una flota de aviones. Para esto, inicialmente se realiza una evaluación de probabilidad de tolerancia al daño en la estructura, mediante el software de acceso libre SMART|DT[10]. Posteriormente, teniendo como datos de entrada diferentes tipos de inspecciones, costos e intervalos de tiempo, se plantea el desarrollo de un Algoritmo Genético que permita encontrar una óptima solución a la hora de realizar el cronograma de inspecciones de esta estructura.

# II. ANTECEDENTES

En los primeros años de la industria aeronáutica, los programas de mantenimiento realizados a los aviones se desarrollaban con base a las experiencias individuales de pilotos y mecánicos, por lo que resultaban simples y carentes de confiabilidad. Posteriormente, con el crecimiento de este sector, en la década de 1950 se introdujo el proceso de mantenimiento denominado Hard-Time (HT), en este la aeronave era desmontada y reacondicionada periódicamente, siendo el fabricante el encargado de proporcionar estos cronogramas de mantenimiento[11].

Una década más tarde, la administración federal de aviación (*Federal Aviation Administration*, **FAA)** dio origen a un segundo proceso de mantenimiento definido como On-Condition (OC), en el cual se determinaba que los componentes del avión deben inspeccionarse periódicamente o ser contratados con algún estándar físico apropiado para determinar si este puede continuar o no en servicio. Más adelante, 1968 se creó el Grupo Directivo de Mantenimiento (MSG), el cual planteó un proceso lógico de decisión para el desarrollo del programa de mantenimiento de una aeronave, esto fue definido en el documento denominado "MSG-1 - Evaluación de mantenimiento y desarrollo de programas".

En 1970, surge el MSG-2 como una actualización del MSG-1, en esta metodología se introduce el concepto de Condition Monitoring (CM), el cual sugiere el monitoreo y análisis del rendimiento mecánico de ciertos componentes del avión, con el fin de comparar los resultados con niveles operativos estándar, esto se ideó con el objeto de eliminar las inspecciones periódicas en algunos elementos estructurales del avión. A partir de esta década las tareas de mantenimiento fueron derivando de uno de los tres tipos proceso; HardTime, On-Condition, Condition Monitoring o alguna combinación de los tres. En 1979, surge el MSG-3, este proporciona un enfoque inteligente del mantenimiento, seleccionando tareas más efectivas y eliminando así tareas de mantenimiento innecesarias. Actualmente, esta es la única metodología aceptada por las autoridades de la aviación[12].

El MSG-3 organiza las tareas de mantenimiento realizando un análisis separado para las siguientes cuatro secciones;

- Sistemas y centrales eléctricas
- Inspecciones zonales
- Inspecciones estructurales
- Rayos /Radiofrecuencia de Alta Intensidad (L/HIRF)

Donde cada sección dentro el MSG-3 contiene una metodología y procesos lógicos específicos.

En particular, en el MSG-3 el Programa de Inspección Estructural, consiste básicamente en una estrategia que permite detectar y reparar de manera oportuna los daños estructurales de una aeronave, para esto plantea el análisis de la estructura por elementos significativos estructurales (Structural Significant Items, SSI) y posteriormente dentro de estos se identifican los elementos estructurales principales (PSE), también conocidos como ubicaciones críticas de fatiga/fractura ( Fatigue/Fracture Critical Locations, FCL), estos elementos pueden ser identificados debido a que contribuyen significativamente al soporte de cargas en vuelo, tierra o de presurización, y cuya integridad es esencial para mantener la integridad general del avión.

Generalmente, los PSE son examinados en busca de probabilidad de daño accidental, deterioro ambiental o daño asociado a la fatiga, mediante procedimientos de ensayos visuales y/o no destructivos[13]. En esta investigación se busca crear un cronograma de mantenimiento para un PSE, el cual se ajuste a la progresión de las grietas en la estructura, evitando así mantenimientos innecesarios.

# III. OBJETIVOS

## *A. Objetivo general*

Desarrollar un Algoritmo Genético que basado en restricciones de riesgo y costo permita optimizar un calendario de inspecciones para una flota de aviones.

## *B. Objetivos específicos*

a. Investigar referentes de Algoritmos Genéticos que puedan ser homologables para optimizar un calendario de inspecciones de una flota de aviones.
b. Implementar un Algoritmo Genético permita optimizar un calendario de inspecciones de una aeronave y una flota de aviones.
c. Evaluar la efectividad del algoritmo genético desarrollado.

# IV. MARCO TEÓRICO

## *A. Conceptos estructurales*

### *1) Tolerancia al daño*

En el del campo de la ingeniería aeronáutica, el concepto de tolerancia al daño es muy utilizado como criterio de diseño, este hace referencia a la capacidad de las estructuras para soportar la carga de diseño y mantener su función en presencia de grietas y otros tipos de daños[14].

### *2) Falla por fatiga*

La falla por fatiga hace referencia a la formación y propagación de grietas dentro de una estructura debido a una carga repetitiva, esta última, en la gran mayoría de los casos está por debajo de los esfuerzos de fluencia del material. En este contexto, la naturaleza cíclica de la carga que hace que las imperfecciones microscópicas propias de los materiales, se conviertan en una grieta[15]. Cabe resaltar que de falla asociada a esto, se define cuando el crecimiento de la grieta dado por fatiga alcanza el tamaño crítico[16]

### *3) Crecimiento de grietas*

Durante el desarrollo de la fase de crecimiento de grieta en estructuras dúctiles, se presentan tres etapas:

- Nucleación: en esta etapa una o más grietas comienzan a desarrollarse, estas generalmente tienden a aparecer en aquellas zonas donde se tiene una fuerte concentración de esfuerzos. Esta etapa a veces no se da, esto es debido a que la estructura puede contener imperfecciones tipo grieta por defecto.
- Crecimiento: periodo en que algunas o todas las grietas crecen en la zona plástica donde se originó, debido a los esfuerzos continuos al que es sometido la estructura.

- Propagación: crecimiento de una o todas las grieta fuera del campo de concentración de esfuerzos donde estas se originaron, hasta producir el fallo final[17].

La Figura 1 muestra la curva característica para el proceso de propagación de grietas dentro de un material dúctil, en esta se pueden observar las tres etapas anteriormente descritas.

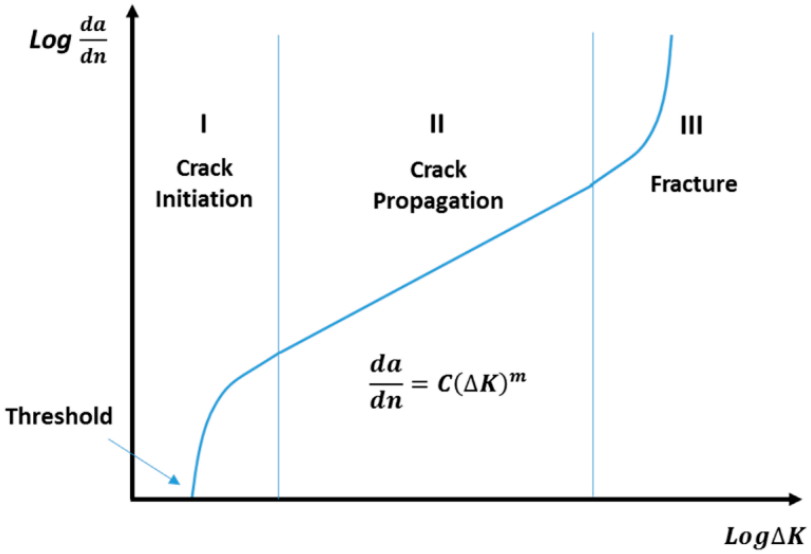



**Fig. 1.** Curva característica de crecimiento de grieta[17]

Donde la variable $a$ hace referencia a la longitud de la grieta y expresión $\frac{da}{dn}$ se refiere al crecimiento de grieta asociado a un n ciclo de cargas. Por otro lado, $\Delta K$ es el promedio dado por la intensidad de tensión máxima y mínima durante el ciclo[17].

*4) Análisis probabilístico de tolerancia al daño*

El análisis probabilístico de tolerancia al daño (*Probabilistic Damage Tolerance Analysis* - PDTA) consiste en términos muy generales, en la ejecución de modelos de crecimiento de grietas para calcular el tamaño de estas y la resistencia residual de la estructura en función del tiempo, esto con el objetivo de estudiar la probabilidad de falla, probabilidad acumulada de falla y tasa de riesgo de una estructura. El análisis probabilístico de tolerancia al daño (PDTA) implica la evaluación del crecimiento de grietas por fatiga, a través de un análisis combinado de

crecimiento determinista de grietas, métodos probabilísticos y modelado de variables aleatorias[18].

*5) Probabilidad de falla*

La probabilidad de falla (*Probability Of Failure – POF*) durante vuelo puede ser conceptualizada como la probabilidad de que la carga máxima experimentada durante el vuelo exceda la resistencia residual de la estructura. Esto, siempre y cuando no haya un registro de fallas antes de ese vuelo[19]. La ecuación 1, resume matemáticamente lo anteriormente expuesto.

$$POF(t) = P[\sigma_{max} > \sigma_{rs}(t)] \qquad (1)$$

Donde:

$POF(t)$ es la probabilidad de falla a través del tiempo.

$\sigma_{max}$ es la carga máxima durante el vuelo.

$\sigma_{rs}(t)$ es la resistencia residual de la estructura.

*6) Función de riesgo*

La función de riesgo instantáneo o tasa de riesgo (Hazard Function), también denotada por la tasa de falla instantánea $\lambda(t)$, es utilizada para cuantificar la probabilidad de que un componente falle en el siguiente intervalo de tiempo infinitesimal $\Delta t$ [20].

*7) Ensayos no destructivos*

Los ensayos no destructivos (*nondestructive testing*, NDT) proporcionan datos sobre el crecimiento de grietas dadas por fatiga sin dañar la estructura. Las técnicas de NDT más comunes en estructuras civiles son la inspección visual (*Visual Testing*, VT), inspección de líquidos penetrantes (*Penetrant Testing*, PT), la inspección de partículas magnéticas (*Magnetic Particle Testing*, MT), la corriente de Eddy (*Eddy Current*, EC), la inspección radiográfica (*Radiographic*

*testing*, RT), la inspección ultrasónica (*Ultrasonic testing*, UT) y la emisión acústica (*Acoustic emisión*, AE)[21].

### *8) Probabilidad de detección*

La probabilidad de detección POD expresa la probabilidad de detectar un defecto de un tamaño dado *a* mediante un determinado tipo de NDT. En general, los resultados de NDT están sujetos a muchos factores aleatorios, tales como la condición de la estructura que se inspecciona y su entorno de servicio, la sensibilidad del equipo de inspección, las imperfecciones materiales y la capacitación y las habilidades del operador. No obstante, es de conocimiento general que la POD aumenta al aumentar el tamaño del defecto[21]. La figura 2 muestra un ejemplo de la curva de la probabilidad de detección para un ensayo de corrientes de Eddy con una distribución Lognormal, un promedio de detección de tamaño de grieta 0.06 in y una desviación estándar de 0.07 una distribución[22].

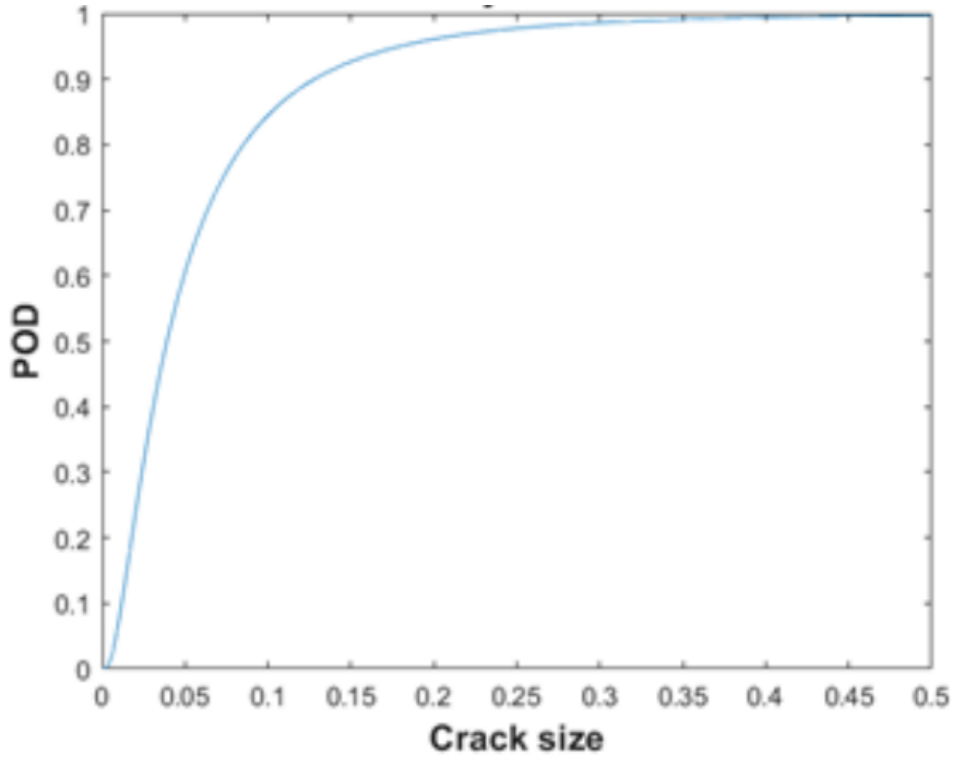



**Fig. 2.** Probabilidad de detección para un ensayo EC[22]

## *B. Conceptos de programación*

### *1) Algoritmo Genético*

Una heurística muy utilizada en la optimización de procesos, son los Algoritmos Genéticos (AG), estos se caracterizan por aplicar un conjunto de métodos inspirados en la selección natural. Los AG son básicamente algoritmos de búsqueda basado en la población que utilizan el concepto de supervivencia del más apto para hallar soluciones a un determinado problema[23].

El potencial de los AG como técnica de optimización, está dado por su simplicidad, facilidad de operación, requerimientos computacionales mínimos y a que permiten una perspectiva global del problema[24]. Una de las mayores ventajas que presenta la implementación de los AG en problemas de ingeniería, es que estos no evalúan todas las soluciones posibles para un problema determinado, por lo que el costo computacional se ve considerablemente reducido cuando se tienen problemas con numerosas posibles soluciones, esto debido a la naturaleza evolucionaria del algoritmo, la cual tiende a descartar soluciones poco viables[25].

El funcionamiento de un algoritmo genético a grandes rasgos puede ser descrito de la siguiente manera; dado un problema específico en el que se requiera encontrar mínimos o máximos en la ejecución de un proceso, se plantea un conjunto inicial de soluciones a este problema, y así mismo una función de aptitud que permita evaluar cuantitativamente a cada potencial solución. Cada una de las soluciones planteadas es evaluada por la función de aptitud, la cual estima qué tan “buena” es una solución con respecto a las demás, permitiendo así descartar aquellas posibilidades que no presentan un buen desempeño. Los arreglos de soluciones más aptos se someten a acciones aleatorias semejantes a las que actúan en la evolución biológica, tales como mutaciones y recombinaciones genéticas, y así el sistema itera hasta encontrar un “individuo” o solución idónea[26]. El esquema de este proceso es presentado en la Figura 3.

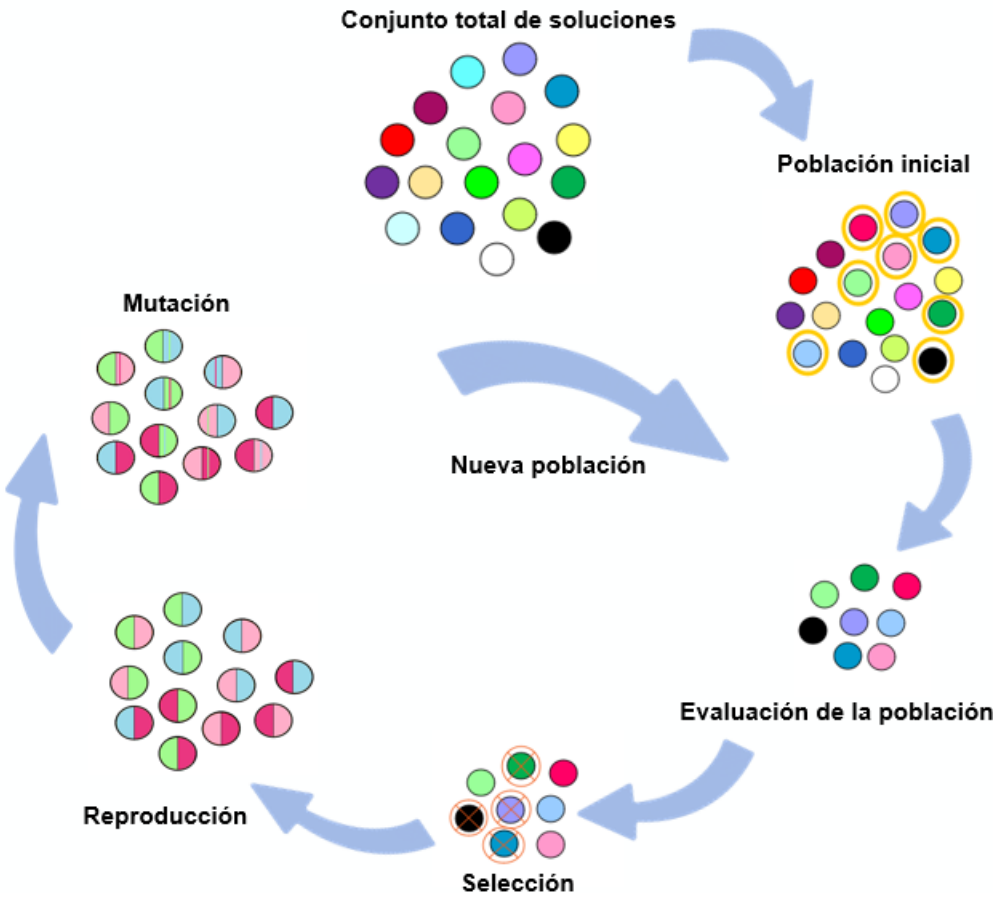



**Fig. 3**. Esquema de Algoritmo Genético.

### *C. Smart|DT*

SMART|DT es software de acceso libre mediante el cual se puede calcular el riesgo asociado al crecimiento de grietas en un elemento estructural de un avión. A grandes rasgos, el programa, a través de un análisis probabilístico realizado con base a la carga máxima, la tenacidad a la fractura y un tamaño de grieta inicial, simula el riesgo asociado a la propagación de esta última en la estructura. Adicionalmente, Smart|DT también proporciona un mecanismo en el que las operaciones de inspección y mantenimiento se pueden incluir en el análisis[27].

# V. METODOLOGÍA

Dentro de esta investigación se pretende realizar un AG con el objetivo de encontrar un equilibrio entre la relación costo-seguridad en el mantenimiento estructural de una aeronave y una flota. Para estudiar el riesgo y probabilidad de falla del avión asociada al crecimiento de grieta, se utiliza el programa SMART|DT, el cual emplea un enfoque PDTA, con el fin de abarcar las grandes incertidumbres que pueden surgir dada la naturaleza del problema y las condiciones críticas que se pueden presentar en algunos aviones.

En la construcción del cronograma de mantenimiento, tanto para el avión como para la flota, se estudia el tipo de inspección a aplicar y la frecuencia de aplicación de estas. Las inspecciones a tener en cuenta en este estudio son; Inspección automatizada de corrientes de Eddy, Inspección de corrientes de Eddy por sonda, Inspección Visual y no aplicar inspección.

El análisis de diferentes opciones de inspección a diferentes intervalos de tiempo abre un amplio espectro de posibles soluciones, en esta investigación el estudio de todas estas posibilidades se realiza mediante un AG. Para la formulación de este, inicialmente se codifican las inspecciones como genes, la tabla I esquematiza lo anteriormente planteado.

TABLA I. CODIFICACIÓN DE INSPECCIONES

| GEN | INSPECCIONES |
|---|---|
| 0 | No inspección |
| 1 | inspección 1 |
| 2 | inspección 2 |
| …. | …..... |
| N | inspección n |

A la combinación de estos genes en los intervalos de tiempo determinados para realizar el análisis, se les domina individuos. A modo de ejemplo, en la tabla II, para el Individuo A, en el tiempo uno no le corresponde inspección, en el tiempo 2 y n le corresponde una inspección tipo 1 y en el tiempo 3 una inspección tipo 2.

TABLA II. INDIVIDUOS FORMADOS A PARTIR DE LA COMBINACIÓN DE INSPECCIONES

| Individuos | Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo n |
|---|---|---|---|---|
| **Individuo A** | 0 | 1 | 2 | 1 |
| **Individuo B** | 2 | 0 | 3 | 0 |
| **…..** | ……. | ……. | ……. | ……. |
| **Individuo n** | 3 | 0 | 0 | N |

Para evaluar que individuos presentan un mejor desempeño en comparación con otros, se mide el desempeño de la variable costo, para esto se utiliza la siguiente expresión $\sum_i^n x_i$ en esta *n* hace referencia al número de intervalos de tiempo seleccionados para realizar el cronograma de inspección y *x* toma el valor del costo asociado al tipo de inspección escogida para el intervalo *i*.

Para el ejercicio de estudio los genes son codificados como se observa en la tabla III. Una vez cifradas las variables y funciones necesarias para el desarrollo del algoritmo genético, se procede a la construcción de este, la metodología utilizada se describe a continuación y es esquematizada en la Figura 4.

TABLA III. CODIFICACIÓN DE LAS INSPECCIONES PARA LA INVESTIGACIÓN

| GEN | INSPECCIONES |
|---|---|
| 0 | No inspección |
| 1 | Visual |
| 2 | Corrientes de Eddy por sonda |
| 3 | Corrientes de Eddy automatizada |

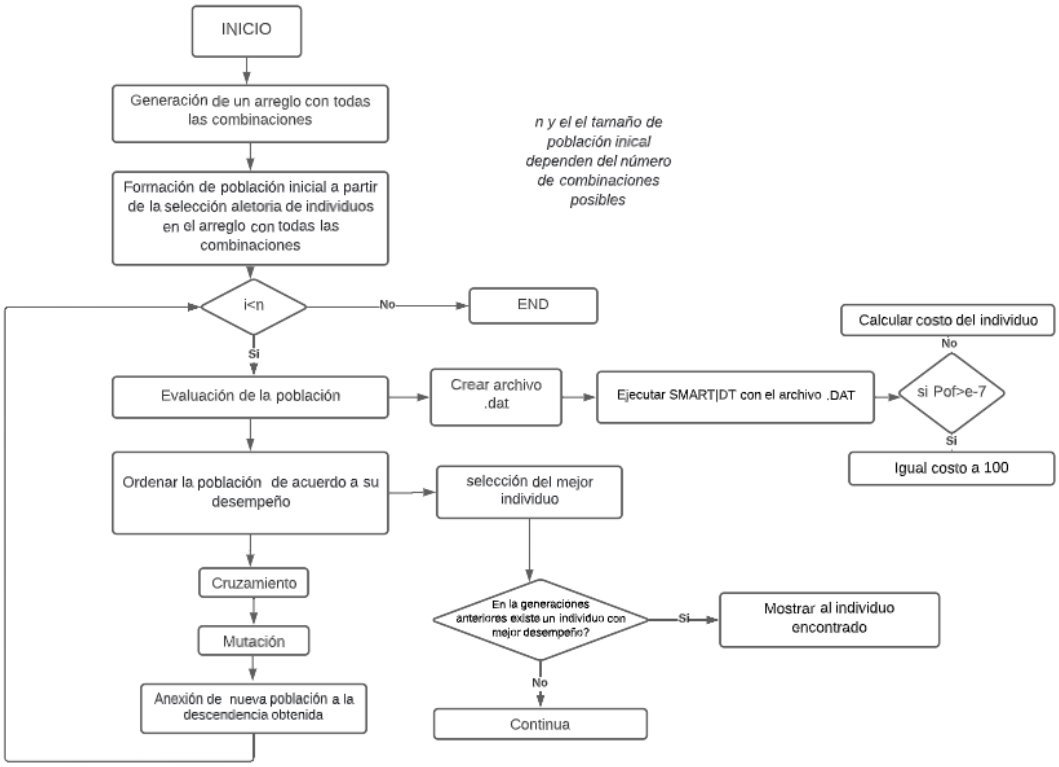



**Fig. 4.** Esquema de algoritmo desarrollado

Inicialmente se crea una matriz que contenga todas las posibles combinaciones de inspecciones, esta se genera mediante un sistema de numeración posicional, ya que este tipo de notación permite construir todas las posibilidades dentro de un conjunto de combinaciones. La generación de cada valor dentro de este arreglo es determinada por la base y en el caso del ejercicio de estudio, esta a su vez es determinada por el número de inspecciones posibles. La tabla IV muestra cómo se estructuran las combinaciones.

TABLA IV. POSIBILIDADES DE INSPECCIÓN

| Indv. | Tiempos de inspección | | | | | | |
|---|---|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_i$ | ….. | ….. | $t_n$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| … | … | … | … | … | … | … | … |
| X | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Luego de que se genera la matriz con todas las combinaciones posibles, se crea una población inicial con un *x* número de "personas", esta se forma a partir de la selección aleatoria de individuos en el arreglo con todas las opciones. La tabla V esquematiza lo anteriormente expuesto.

TABLA V. ESQUEMA DE SELECCIÓN DE POBLACIÓN INICIAL

| Indv. | Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo 4 | Tiempo 5 | Tiempo 6 | Tiempo n |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| … | … | … | … | … | … | … | … |
| n | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Posteriormente esta población entra en un ciclo que itera *n* veces. Es preciso señalar que tanto el tamaño de la población inicial, *x*, y el número de iteraciones, *n*, están directamente relacionados y dependerán del número de opciones posibles. Es así como, a mayor número de soluciones posibles, mayor será el valor de estas variables.

Una vez en el bucle, cada individuo es decodificado y a partir de la información que proporciona, se crea un archivo .DAT, el cual se ejecuta en SMART|DT. Posteriormente, en los resultados arrojados por el programa se evalúa si el riesgo, en el caso de la aeronave y probabilidad de falla, en el estudio de la flota, es menor a $10^{-7}$, en caso de ser así, se calcula el costo asociado al individuo, mediante el uso de la ecuación 1 y en caso contrario, se le asigna un valor de costo de 100, tal como se observa en la Figura 5.

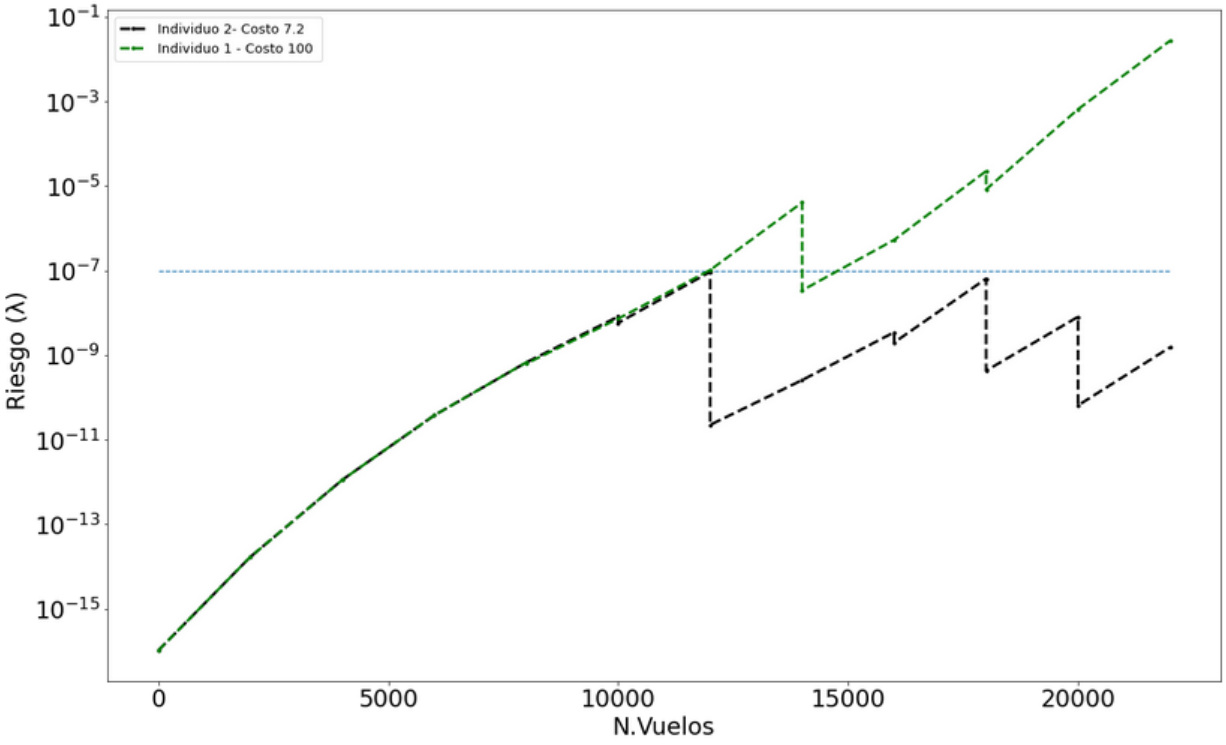



**Fig. 5**. Desempeño de algunas inspecciones

Luego que se han evaluado todas las combinaciones que representan a la población inicial y se tiene un valor de costos asociado a cada una de ellas, se ordena la población en función de este, tal y como se muestra en la tabla VI.

TABLA VI. POBLACIÓN ORDENADA EN BASE AL COSTO Y DESEMPEÑO

| Indv. | Tiempos de inspección [N. Vuelo] | | | | | | | Costo |
|---|---|---|---|---|---|---|---|---|
| | Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo 4 | Tiempo 5 | Tiempo 6 | Tiempo n | |
| 50 | 2 | 0 | 1 | 0 | 3 | 1 | 0 | 7.2 |
| 30 | 2 | 3 | 0 | 2 | 0 | 1 | 0 | 10 |
| 15 | 2 | 0 | 3 | 0 | 0 | 1 | 0 | 12 |
| 5 | 0 | 1 | 2 | 3 | 0 | 0 | 0 | 24 |
| 9 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 100 |
| … | … | … | … | … | … | … | … | … |
| n | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 100 |

Después, se escoge la tercera parte de esta población con mejor desempeño, es decir aquellas que poseen un menor costo. Durante el proceso de selección también se escoge al individuo que mejor desempeño tuvo, en el caso de que sea la primera iteración se muestra en pantalla la combinación que representa este sujeto, en caso de que n sea mayor que uno, se procede a realizar una comparación con el mejor resultado obtenido en las generaciones anteriores y si el costo asociado a este es mejor, se muestra el nuevo ente con mejor desempeño, en caso contrario el algoritmo continua con siguiente proceso.

En el proceso de cruzamiento se utiliza un método denominado cruce de N puntos, en este se seleccionan múltiples puntos de cruce a lo largo de las cadenas que representan a los padres y se intercambian los bits que están muy próximos a los sitios cruzados. En la tabla VII, se muestra un ejemplo de cruzamiento para un N igual 1. Se debe tener en cuenta un N muy grande a veces reduce el rendimiento del algoritmo genético[28].

TABLA VII. PROCESO DE CRUZAMIENTO

| Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|
| Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo 4 | Tiempo 5 | Tiempo 6 | Tiempo n |
| 2 | 3 | 0 | 0 | 3 | 1 | 0 |
| 2 | 0 | 1 | 2 | 0 | 1 | 0 |
| … | … | … | … | … | … | … |

Una vez la descendencia ha sido obtenida, en esta se realiza un proceso de mutación, esto con el fin de mantener siempre la variabilidad genética de la población y así evitar que el algoritmo caiga muy a menudo en óptimos locales[29]. Lo anterior, se lleva a cabo en este estudio, mediante la selección aleatoria de un gen en la cadena que representa al individuo, este gen es redefinido con un nuevo valor seleccionado aleatoriamente. La tabla VIII muestra el proceso anteriormente descrito para la descendencia obtenida durante el cruzamiento mostrado en la tabla VII.

TABLA VIII. PROCESO DE MUTACIÓN

| Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|
| Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo 4 | Tiempo 5 | Tiempo 6 | Tiempo n |
| 2 | 2 | 0 | 0 | 3 | 1 | 0 |
| 2 | 0 | 1 | 2 | 0 | 0 | 0 |
| … | … | … | … | … | … | … |

Cabe resaltar, que para esta investigación un 80% de los descendientes será seleccionado al azar para ser expuesto a esta transformación.

Una vez ha finalizado el proceso de mutación y se tiene la descendencia creada, a esta se le anexa un nuevo arreglo de población creado de la misma forma en la que se procedió con la matriz de la población inicial, esto se esquematiza en la tabla IX. Lo anterior, con el fin de extender la variabilidad genética y acelerar el proceso de encontrar respuestas optimas. Finalmente, esta población representa la nueva generación a ser evaluada y el proceso se repite múltiples veces.

TABLA IX. NUEVA GENERACIÓN

| Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|
| Tiempo 1 | Tiempo 2 | Tiempo 3 | Tiempo 4 | Tiempo 5 | Tiempo 6 | Tiempo n |
| 2 | 2 | 0 | 0 | 3 | 1 | 0 |
| 2 | 0 | 1 | 2 | 0 | 0 | 0 |
| … | … | … | … | … | … | … |
| 0 | 1 | 0 | 0 | 0 | 3 | 0 |
| 0 | 0 | 1 | 1 | 3 | 2 | 0 |

Todo el proceso anteriormente descrito, fue programado en Python, dadas las facilidades que presenta este lenguaje para conectar con programas externos y para la manipulación de datos.

# VI. RESULTADOS

El objetivo principal de esta investigación es realizar un Algoritmo Genético que permita la programación de un calendario de inspecciones para un PSE, en el que la relación costo-seguridad se encuentre equilibrada. Por lo anterior, la metodología previamente descrita es empleada para optimizar el calendario de inspecciones del larguero de las alas de un jet corporativo, el cual es un elemento estructural clasificado como PSE. La importancia de este elemento es debido a que el ala del avión está expuesta a todo tipo de cargas aerodinámicas y la estructura principal que soporta estas cargas son los largueros, los cuales son básicamente vigas que se extienden transversalmente a través del ala. El esquema de la estructura de estudio es mostrado en la Figura 6.

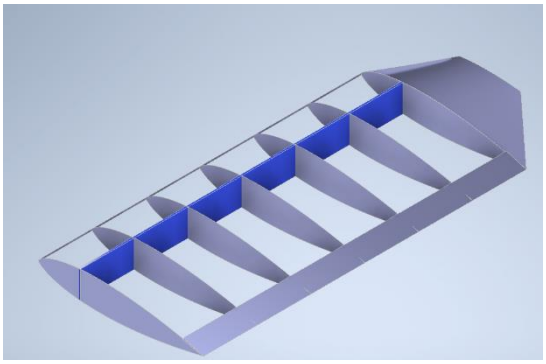


**Fig. 6.** Esquema del alerón

Debido a que larguero es integrado al revestimiento y a las nervaduras por medio de fijación mecánica, generalmente se tiene una fuerte concentración de tensión en algunos orificios de sujeción, lo que puede dar como resultado el inicio de grietas por fatiga[30]. Por lo anterior, en esta investigación, el análisis de crecimiento de grieta se realiza en los agujeros de los orificios de los pernos del larguero.

Para el análisis del crecimiento de grieta en los orificios de los pernos del larguero se realizan las siguientes simplificaciones geométricas. Dado que esta estructura es generalmente diseñada como una viga de perfil en forma de I, evocando al principio de simetría, solo se realizará el análisis en una cuarta parte de la estructura. Por parte, debido a que el crecimiento de

grieta se concentra en el patín del perfil, se analiza una geometría idealizada en la que solo se abarca este. La Figura 7 muestra la simplificación de la geometría anteriormente descrita

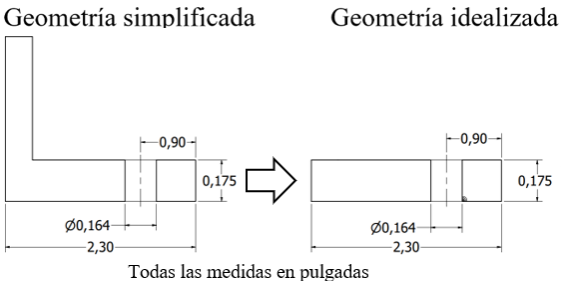



**Fig. 7**. Simplificaciones geométricas.

La tabla X resume los parámetros de entrada asociados a la estructura de estudio proporcionados al programa SMART|DT. En esta se proporciona un tamaño de grieta o defecto inicial, así como también las propiedades mecánicas del material y un promedio del espectro de cargas al que es sometida la estructura durante el servicio. Estos valores son introducidos al software por medio de un archivo .DAT

TABLA X. PARÁMETROS ESTRUCTURALES DE ENTRADA

| Variable | Distribución | Prom. | Des. Estándar | Notas |
|---|---|---|---|---|
| Tamaño de grieta inicial | Log-normal | 9.055e-3 in | 0.001252 in | Orificio de sujeción escariado |
| Tenacidad a la fractura | Normal | 37.0 ksi | 3.8 ksi | 7050-T651 Plate |
| Módulo de cedencia | Normal | 57 ksi | 0.0 ksi | N.A. |
| AVE | Gumbel | 16.74 ksi | 2.08 ksi | N.A. |

La tabla XI muestra las distribuciones de probabilidad de detección de grietas para cada una de las inspecciones a tener en cuenta al realizar el cronograma. Cabe mencionar que en este estudio se asume que las grietas encontradas por medio de estos chequeos son totalmente reparadas. Adicionalmente, a cada una de estas se les asigna un valor de costo representativo el cual está directamente relacionado con el nivel de complejidad que conlleva su aplicación.

TABLA XI. PARÁMETROS DE INSPECCIONES

| Tipo de inspección | Distribución | Prom. [in] | Des. Estándar [in] | Costo |
|---|---|---|---|---|
| Visual | Lognormal | 0.99714 | 3.66907 | 0.1 |
| Corrientes de Eddy por sonda | Lognormal | 0.0788 | 0.0302 | 1 |
| Corrientes de Eddy para agujero de pernos automatizada | Lognormal | 0.018 | 0.0109 | 5 |

Para realizar el cronograma de inspecciones en la estructura anteriormente descrita, se toman dos escenarios de estudio; en el primero se realiza la construcción del calendario de inspecciones para una aeronave y en el segundo se desarrolla el calendario para una flota de 4 aviones.

Para el primer escenario de análisis (calendario de inspecciones para una aeronave), adicionalmente a la estructura ya mencionada, se toma en cuenta otro caso estudio. En este se examina el riesgo de falla de un ejemplo académico tomado del manual, *Damage Tolerant Design Handbook*[31]. La tabla XII muestra los valores estructurales tomados como variables de entrada para SMARR|DT.

TABLA XII. PARAMETROS DE ENTRADA ESTRUCTURALES (CASO II)

| Variable | Distribución | Prom. | Des. estándar | Notas |
|---|---|---|---|---|
| Tamaño de grieta inicial | Log-normal | 0.00248 in | 0.00129 | Orificio de sujeción escariado |
| Tenacidad a la fractura | Normal | 26.0 ksi | 2.0 | 7050-T651 Plate |
| Módulo de cedencia | Normal | 80 ksi | 0.0 | N.A. |
| AVE | Gumbel | 14.5 ksi | 008 | N.A. |

La tabla XIII muestra las distribuciones de probabilidad de detección de grietas para cada una de las inspecciones a utilizar en este caso y el valor asociado a cada una de ellas.

TABLA XIII. PARÁMETROS DE INSPECCIONES

| Tipo de inspección | Distribución | Prom. [in] | Des. estándar | Costo |
|---|---|---|---|---|
| Visual | Lognormal | 1.05964 | 3.66907 | 0.1 |
| Corrientes de Eddy por sonda | Lognormal | 0.1413 | 0.0302 | 1 |
| Corrientes de Eddy para agujero de pernos automatizada | Lognormal | 0.0180 | 0.0109 | 5 |

A continuación, se describe la aplicación para los escenarios anteriormente plantados (la optimización del calendario de inspecciones para una aeronave y el calendario para una flota de 4 aviones), y se detallan los resultados obtenidos para estos.

## *A. Optimización del calendario de inspecciones para una aeronave*

Como se mencionó anteriormente, para optimizar el calendario de inspecciones de una aeronave, se tuvieron en cuenta dos casos de estudio. El primer caso, correspondiente a un alerón de las alas de un jet y el segundo a un ejemplo académico. En breve se describe cada uno de estos.

### *a) Optimización del calendario de inspecciones de un alerón*

Para realizar el calendario de inspecciones para el alerón, en principio, los datos estructurales del larguero mostrados en la tabla X son computados por SMAR|DT, es así como en la Figura 8 se observa el riesgo de falla asociado al crecimiento de grieta y a la reducción en la resistencia residual en el larguero durante un periodo de 30000 vuelos. Adicionalmente, también se grafica el riesgo máximo permitido por las autoridades aeronáuticas ($10^{-7}$) bajo el cual que se debe programar el mantenimiento. Por otro lado, es preciso señalar que cada vuelo tiene una duración en promedio de 1.65 horas.

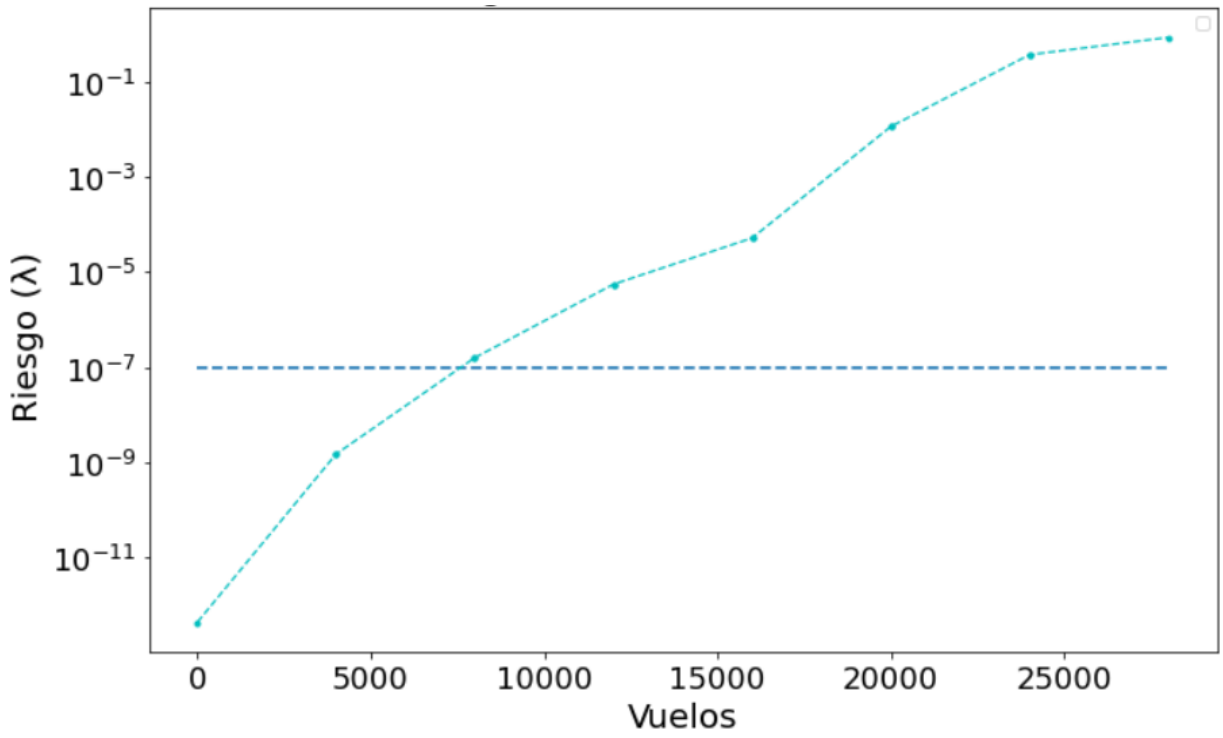



**Fig. 8.** Riesgo de falla del larguero

En la Figura 8, se observa como al no programar ninguna inspección, el umbral de probabilidad de falla se alcanza antes de las 10,000 horas de vuelo. Además, en la Figura 9, se muestra el desempeño de cada una de las inspecciones, mostradas en la tabla XI, a 12000 vuelos.

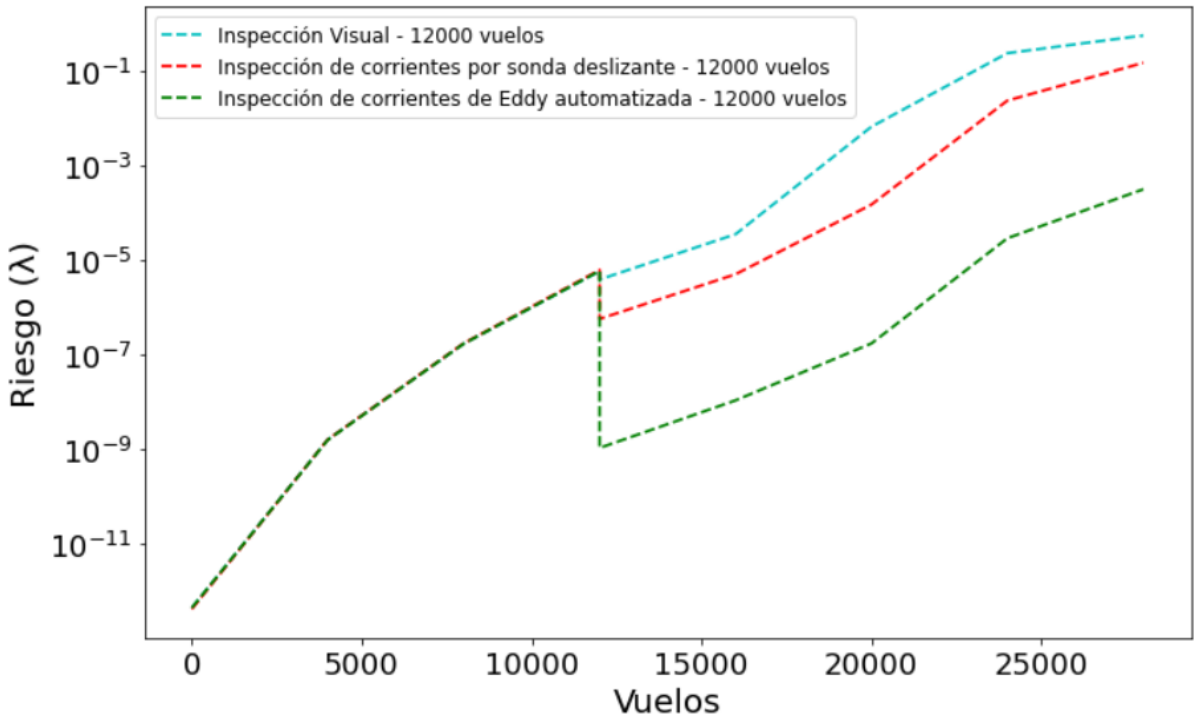
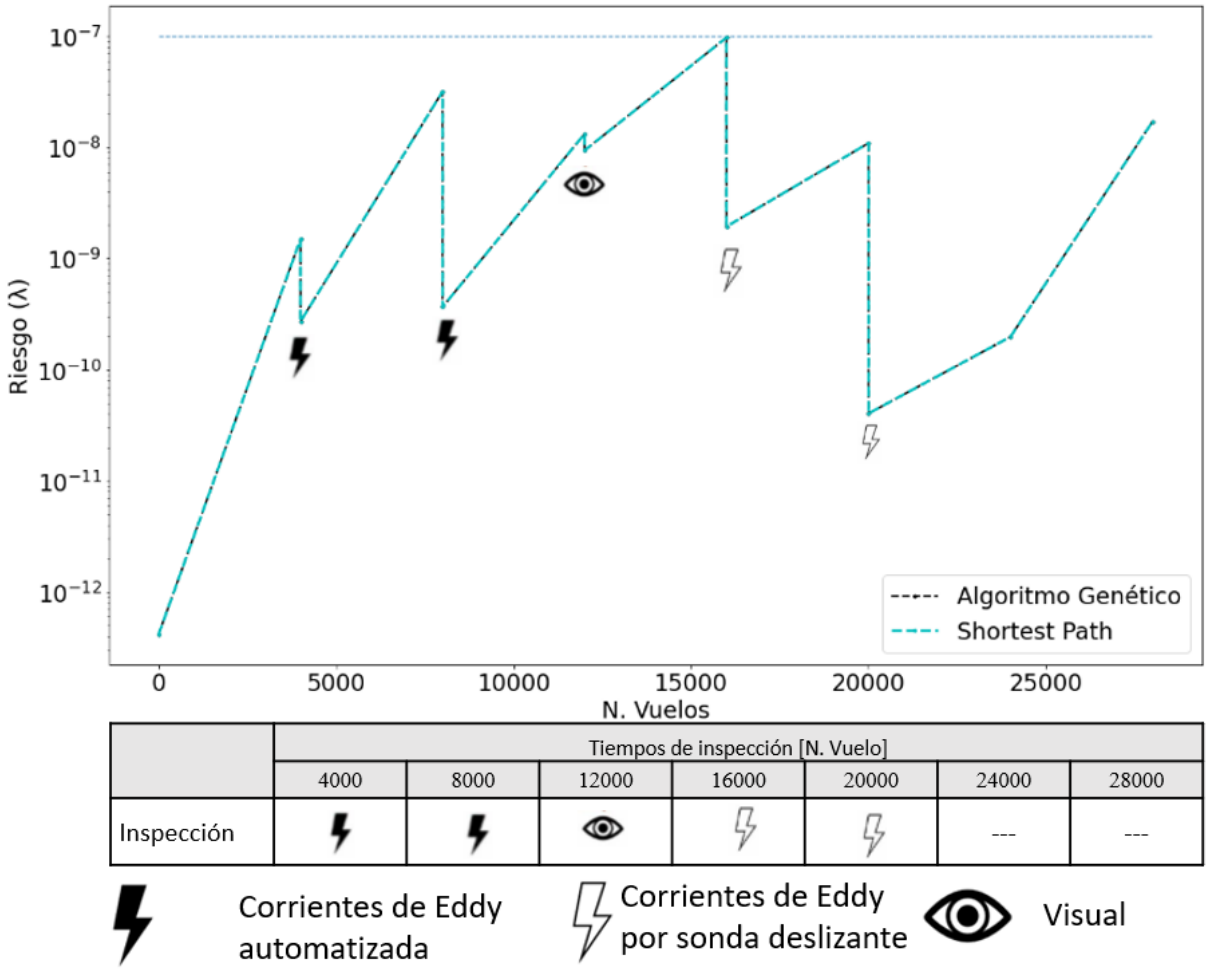



**Fig. 9.** Desempeño de las inspecciones a 12.000 vuelos

Teniendo en cuenta el escenario anteriormente planteado, para realizar el cronograma de mantenimiento de este elemento, se escogen los siguientes siete intervalos de vuelo en los que se estudia la posibilidad de aplicar una inspección: 4.000; 8.000; 12.000; 16.000; 20.000; 24.000; 28.000. Adicionalmente, para las variables referentes al AG se selecciona una población inicial "*x*" de 25 y un numero de iteraciones en las que se repite el ciclo "*n*" de 30. Estos valores fueron escogidos debido a que presentaron buenos resultados respecto a la variable de salida deseada (el cronograma de inspecciones optimizado) y el tiempo que se demora este algoritmo en hallarla.

Los resultados obtenidos para este caso de estudio se muestran en la Figura 10. En esta, se puede observar que las inspecciones a realizar se han esquematizado mediante figuras alusivas al tipo de inspección. Así mismo se ha dispuesto de un cronograma que contiene los intervalos en los que se debe realizar el chequeo y a partir de este se grafica el riesgo de falla de la estructura en función del número vuelo.




| | Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4000 | 8000 | 12000 | 16000 | 20000 | 24000 | 28000 |
| Inspección | ϟ | ϟ | 👁 | ϟ | ϟ | --- | --- |

ϟ Corrientes de Eddy automatizada ϟ Corrientes de Eddy por sonda deslizante 👁 Visual

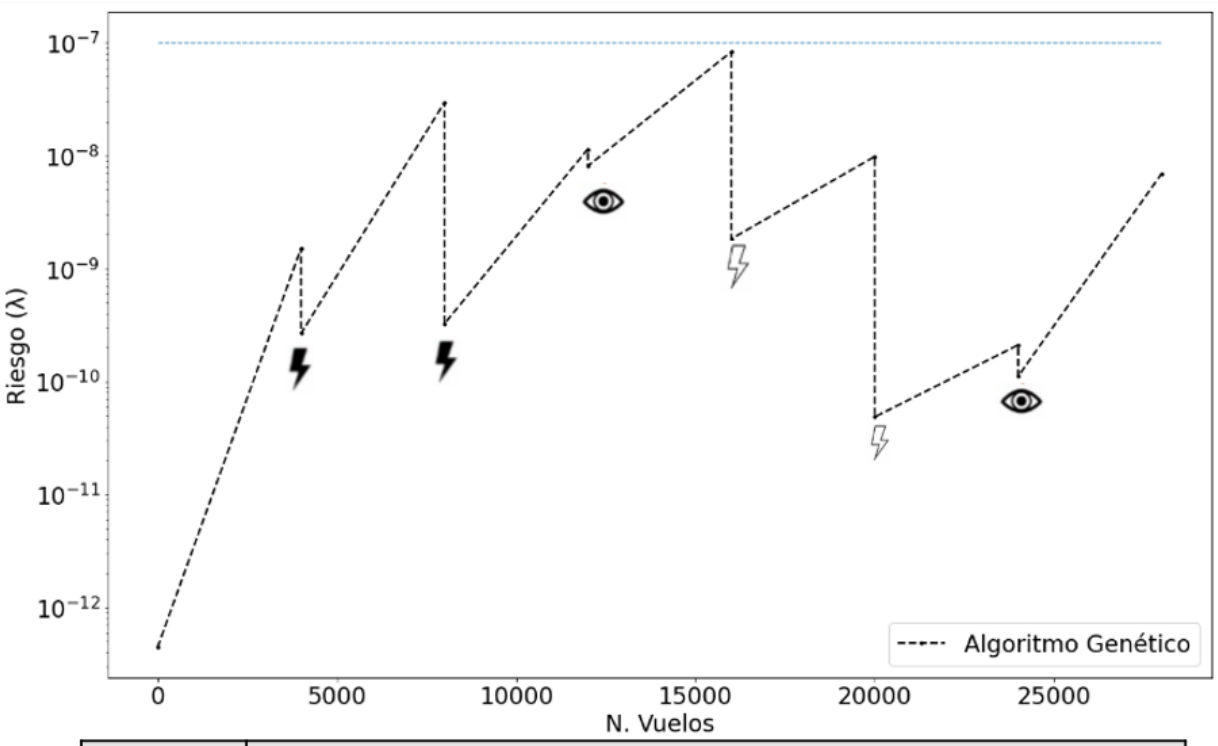
**Fig. 10**. Soluciones halladas por el AG y algoritmo Shortest Path

En los resultados se puede apreciar cómo los intervalos de inspección se adecuan al incremento en el riesgo de falla y por ende a la progresión de las grietas en la estructura, encontrando así un punto de equilibrio entre los costos asociados a la ejecución del mantenimiento y la confiabilidad del elemento estructural.

Los resultados arrojados por el AG son validados por otro algoritmo denominado Shortest Path, el cual es un algoritmo que evalúa todas las combinaciones de inspecciones posibles y selecciona la mejor[32]. En la Figura 10 se puede observar como el AG predice adecuadamente el cronograma de inspecciones, ya que sus resultados se encuentran en total concordancia con la solución planteada por el Shortest Path. Para llegar a esta resolución el AG solo evalúa 1210 de

las 16384 combinaciones y emplea un tiempo aproximado de 525 s, reduciendo así el costo computacional asociado a este análisis.

En este punto, es preciso señalar que debido a su naturaleza aleatoria, el AG no siempre va a evaluar a la misma población y no siempre podrá encontrar la solución óptima. En la figura 11 se observa el resultado dado por el AG para la misma situación mostrada anteriormente. La solución hallada por el algoritmo difiere levemente de la anterior al programar una inspección visual más de la necesaria. Para hallar llegar a esta solución el AG evaluó una población 1287 individuos y empleo un tiempo de 570 s.




| | Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4000 | 8000 | 12000 | 16000 | 20000 | 24000 | 28000 |
| Inspección | ↯ | ↯ | 👁 | ↯ | ↯ | 👁 | --- |

↯ Corrientes de Eddy automatizada ↯ Corrientes de Eddy por sonda deslizante 👁 Visual

**Fig. 11.** Segunda solución hallada por AG

*b) Optimización del calendario de inspecciones ejemplo académico*

Para realizar el calendario de inspecciones para el ejemplo académico, inicialmente, los datos estructurales mostrados en la Tabla XII son computados por SMAR|DT. De esta manera en la Figura 12 se muestra el riesgo de falla asociado al crecimiento de grieta y a la reducción en la resistencia residual del elemento durante un periodo de 22000 vuelos. También se puede observar cómo al no programar ninguna inspección, el umbral de probabilidad de falla se alcanza aproximadamente a las 12.000 horas de vuelo.

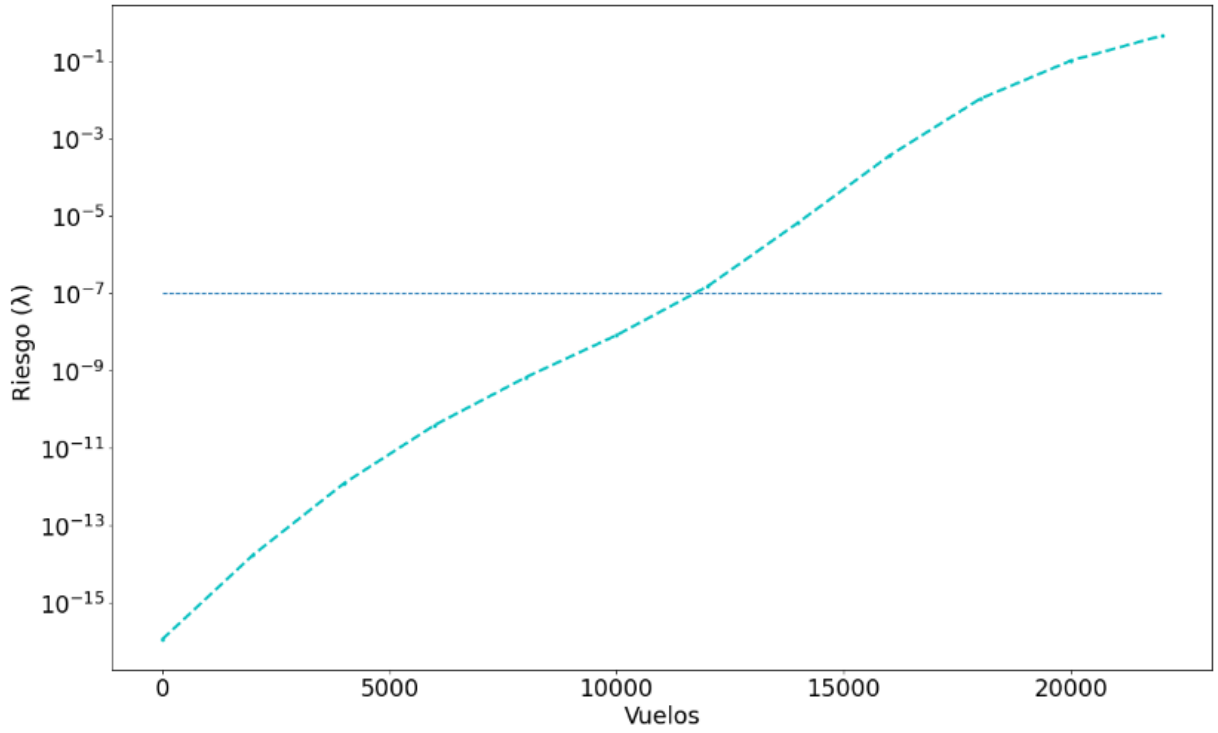



**Fig. 12**. Riesgo de falla para el ejemplo académico

Para realizar el cronograma de mantenimiento de este caso, se escogen los siguientes diez intervalos de vuelo en los que se estudia la posibilidad de aplicar una inspección: 2.000; 4.000; 6.000; 8.000; 10.000; 12.000; 14.000; 16.000; 18.000; 20.000. Por otro lado, para las variables referentes al AG se selecciona una población inicial "*x*" de 40 y un numero de iteraciones en las que se repite el ciclo "*n*" de 50.

Los resultados obtenidos para este caso de estudio se muestran en la Figura 13. En esta se puede observar como el cronograma de mantenimiento recomendado por el AG nuevamente encuentra un punto de equilibrio entre el costo del manteamiento y la confiabilidad estructural de este. Además, se puede apreciar como las predicciones dadas por el AG se encuentran en total concordancia con los resultados obtenidos por el Shortest Path. En este caso, se observa que este último proporciona dos diferentes soluciones que repercuten de manera idéntica en la seguridad de estructura y tienen el mismo costo de aplicación. El AG es capaz de encontrar una de estas evaluando 3324 de las 1,048,576 y empleando un tiempo de 557 s.

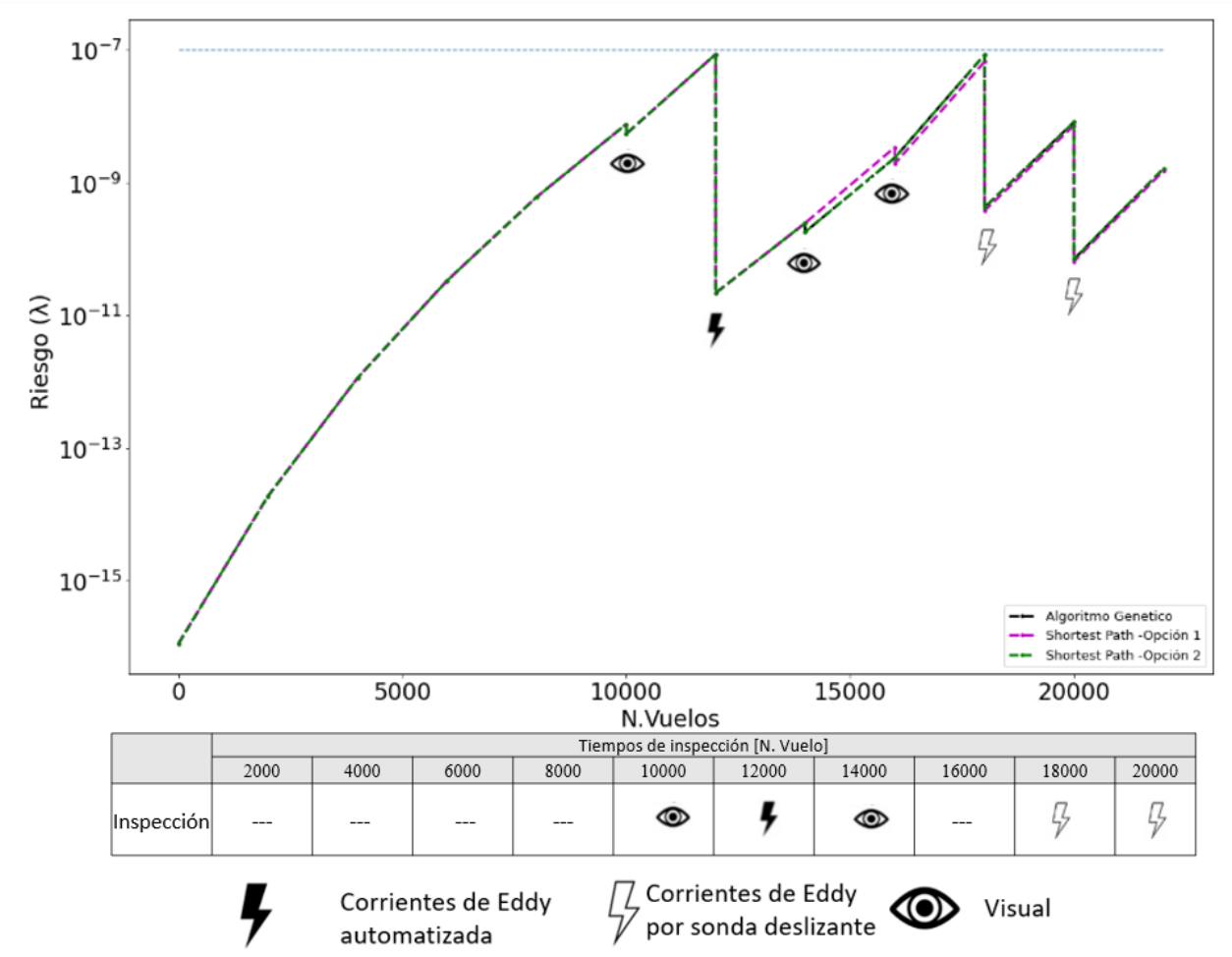



**Fig. 13.** Resultados para ejemplo académico

### *B. Optimización del calendario de inspecciones para una flota*

A continuación, se muestra el desarrollo para la optimización del cronograma de inspecciones para el larguero del ala de una flota de aviones. Para realizar este análisis, se toman los datos estructurales del larguero mostrados en la tabla X, es preciso señalar que bajo este escenario estos datos representarán un comportamiento promedio de este elemento en la flota.

Para esta investigación se tomó una flota de 4 aviones, la tabla XIV especifica los intervalos de vuelo en los que se realiza el análisis de esta.

TABLA XIV. ESPECIFICACIONES DE LA FLOTA DE ESTUDIO

| Aeronave | N. vuelos | N. vuelos previstos |
|---|---|---|
| Aeronave 1 | 1000 | 1000 |
| Aeronave 2 | 2500 | 500 |
| Aeronave 3 | 5000 | 500 |
| Aeronave 4 | 6000 | 500 |

En la Figura 14 se grafican las condiciones de la flota de estudio, mostrada en la tabla XIV.

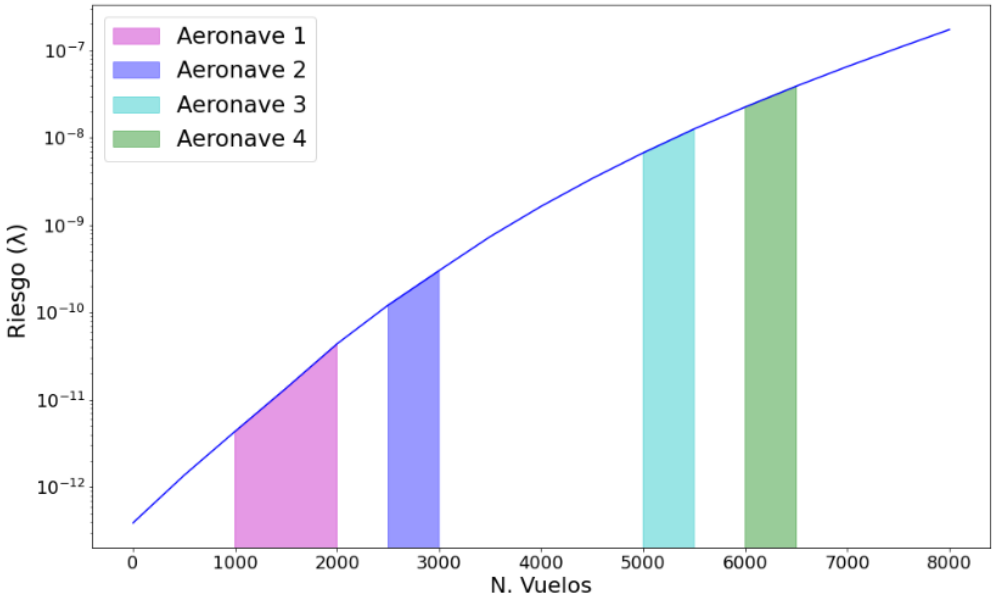
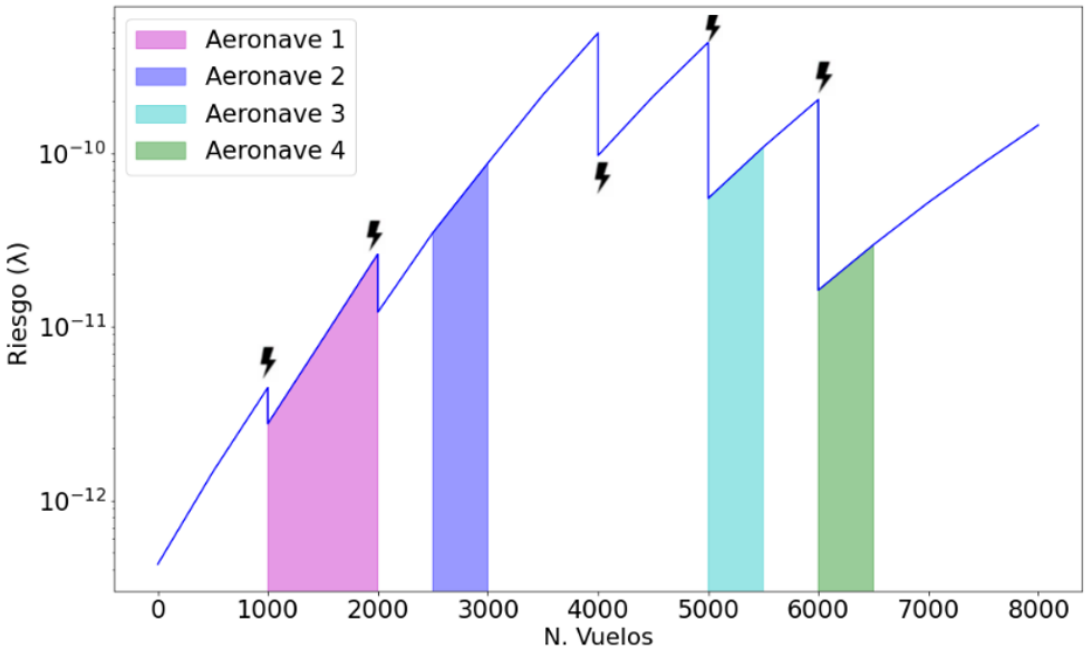



**Fig. 14.** Características de la flota de estudio

Para calcular la POF de la flota, inicialmente se calcula la POF de cada uno de estos aviones en los intervalos de tiempo especificados y se procede a sumar. Para hacer esto se calcula el área bajo la curva de la función de riesgo en el intervalo de tiempo en el que se prevén los vuelos. Es decir, se suman las áreas sombreadas señaladas en la Figura 14.

Para realizar el cronograma de la flota se plantea la posibilidad de aplicar una inspección en los vuelos: 1.000; 2.000; 3.000; 4.000; 5.000; 6.000; 7.000. Adicionalmente, para las variables referentes al AG se selecciona una población inicial, *x*, de 25 y un numero de iteraciones, *n*, de 30.

Los resultados obtenidos para el cronograma de la flota se muestran e n la Figura 15. En esta, el plan de mantenimiento recomendado por el algoritmo prevé que para poder mantener la POF por debajo de $10^{-7}$, se deben asegurar inspecciones en etapas tempranas de los aviones que conforman la flota.




| | Tiempos de inspección [N. Vuelo] | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
| Inspección | ϟ | ϟ | --- | ϟ | ϟ | ϟ | --- |

ϟ Corrientes de Eddy automatizada  ϟ Corrientes de Eddy por sonda deslizante  Visual

**Fig. 15.** Soluciones halladas por el AG

Los resultados del AG fueron nuevamente validos por el algoritmo Shortest Path, encontrando total concordancia entre las soluciones propuestas por estos. Para realizar este análisis el AG evaluó 1570 de las 16384 posibilidades y empleó un tiempo de 653 s.

# VII. CONCLUSIONES

Los resultados muestran como el cronograma de inspección dado por el AG, se adapta a la progresión de la grieta en la estructura y sugiere intervalos variables entre inspecciones, evitando así, la programación de estas cuando no son necesarias o pueden resultar no efectivas para detectar las grietas. Además, los resultados sugieren que la aplicación de SMART|DT en conjunto con un AG, muestra ser una herramienta bastante eficaz a la hora de optimizar cronogramas de mantenimiento para una estructura determinada de un avión y de una flota. Sin embargo, aún se hace necesario realizar más estudios que validen la integración de estas dos herramientas como una metodología óptima para realizar cronogramas aplicables en un ambiente reales.

Por otro lado, también es preciso resaltar la utilidad de los algoritmos genéticos cuando se tienen universos de soluciones muy grandes, ya que, dada la naturaleza aleatoria de estos, se pueden estudiar más ampliamente los espacios solución, permitiendo así, no solo encontrar resoluciones optimas, sino también disminuir el tiempo de análisis. Así mismo, es preciso mencionar que debido a que el AG imita la naturaleza aleatoria del proceso de evolución, este no siempre proporcionara las inspecciones óptimas, aunque si se acercara mucho a estas. Por lo anterior, se requiere el estudio de técnicas que proporcionen siempre la mejor solución sin que esto signifique incrementar demasiado los requerimientos computacionales.

## REFERENCIAS

[1] L. S. Goksel, “fatigue and damage tolerance assessment of aircraft structure under uncertainty”, Georgia Institute of Technology, 2013.

[2] G. Wild, L. Pollock, A. K. Abdelwahab, y J. Murray, “Need for Aerospace Structural Health Monitoring”, *Int. J. Progn. Heal. Manag.*, vol. 12, núm. 3, pp. 1–16, 2021, doi: 10.36001/ijphm.2021.v12i3.2368.

[3] S. M. O. Tavares y P. M. S. T. de Castro, “An overview of fatigue in aircraft structures”, *Fatigue Fract. Eng. Mater. Struct.*, vol. 40, núm. 10, pp. 1510–1529, 2017, doi: 10.1111/ffe.12631.

[4] X. Chen, H. Ren, y C. Bil, “Inspection intervals optimization for aircraft composite structures considering dent damage”, *J. Aircr.*, vol. 51, núm. 1, pp. 303–309, 2014, doi: 10.2514/1.C032377.

[5] S. J. Kim y J. H. Choi, “Comparative Study for Inspection Planning of Aircraft Structural Components”, *Int. J. Aeronaut. Sp. Sci.*, vol. 22, núm. 2, pp. 328–337, 2021, doi: 10.1007/s42405-020-00319-x.

[6] F. Grooteman, “A stochastic approach to determine lifetimes and inspection schemes for aircraft components”, *Int. J. Fatigue*, vol. 30, núm. 1, pp. 138–149, 2008, doi: 10.1016/j.ijfatigue.2007.02.021.

[7] L. Lin, B. Luo, y S. S. Zhong, “Development and application of maintenance decision-making support system for aircraft fleet”, *Adv. Eng. Softw.*, vol. 114, pp. 192–207, 2017, doi: 10.1016/j.advengsoft.2017.07.001.

[8] Y. Wang, C. Gogu, N. Binaud, C. Bes, R. T. Haftka, y N. H. Kim, “Predictive airframe maintenance strategies using model-based prognostics”, *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.*, vol. 232, núm. 6, pp. 690–709, 2018, doi: 10.1177/1748006X18757084.

[9] Q. Deng, B. F. Santos, y R. Curran, “A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization”, *Eur. J. Oper. Res.*, vol. 281, núm. 2, pp. 256–273, 2020, doi: 10.1016/j.ejor.2019.08.025.

[10] Small Aircraft Technology [SMART], “SMART|DT (Version 1.0.227) [Software]” SMART, 2022

[11] S. P. Ackert, “Evaluation & Insights of Commercial Aircraft Maintenance Programs”,

*Basics Aircr. Maint. Programs Financ.*, pp. 1–23, 2010.

[12] A. Shannon, “Aircraft Maintenance Handbook For Financiers”, *Aircr. Monit.*, pp. 1–152, 2018.

[13] A. R. Nikouee y S. Adibnazari, “Implementation of ‘MSG-3’” for Crack Growth Analysis of Aircraft SSI Components”, *J. Aerosp. Sci. Technol.*, vol. Vol. 10, núm. No. 1, pp. 1–15, 2013.

[14] “Fracture processes of aerospace materials”, *Introd. to Aerosp. Mater.*, pp. 428–453, 2012, doi: 10.1533/9780857095152.428.

[15] G. Antaki y R. Gilada, “Design Basis Loads and Qualification”, *Nucl. Power Plant Saf. Mech. Integr.*, pp. 27–102, 2015, doi: 10.1016/B978-0-12-417248-7.00002-3.

[16] Y. Bai y W.-L. Jin, “Probability- and Risk-Based Inspection Planning”, *Mar. Struct. Des.*, pp. 689–705, 2016, doi: 10.1016/B978-0-08-099997-5.00037-X.

[17] A. A. Andrade, W. A. Mosquera, y L. V. Vanegas, “Modelos de crecimiento de grietas por fatig. Models of fatigue crack growth”, *Entre Cienc. e Ing.*, vol. 18, núm. 18, pp. 39–48, 2015, [En línea]. Disponible en: http://www.scielo.org.co/pdf/ecei/v9n18/v9n18a06.pdf

[18] J. Ocampo y H. Millwater, “Probabilistic damage tolerance for small airplanes using a linear-elastic crack growth fracture mechanics surrogate model”, *Collect. Tech. Pap. - AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf.*, pp. 1–16, 2013.

[19] J. Ocampo *et al.*, “Probabilistic damage tolerance for aircraft fleets using the FAA-sponsored SMART | DT Software”, núm. June, pp. 7–9, 2017.

[20] G. F. M. de Souza, A. Caminada Netto, A. H. de Andrade Melani, M. A. de Carvalho Michalski, y R. F. da Silva, “Reliability and maintenance fundamentals”, *Reliab. Anal. Asset Manag. Eng. Syst.*, pp. 9–53, ene. 2022, doi: 10.1016/B978-0-12-823521-8.00005-0.

[21] R. Zheng y B. R. Ellingwood, “Role of non-destructive evaluation in time-dependent reliability analysis”, *Struct. Saf.*, vol. 20, núm. 4, pp. 325–339, 1998, doi: 10.1016/S0167-4730(98)00021-6.

[22] M. I. Vallejo Ciro & M. J. Carvajal Loaiza, “Probabilistic damage tolerance analysis using inspection data from integrated sensors”, Bachelor’s degree project, Mechanical Engineering, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2021.

[23] S. Katoch, S. S. Chauhan, y V. Kumar, *A review on genetic algorithm: past, present, and future*, vol. 80, núm. 5. Multimedia Tools and Applications, 2021. doi: 10.1007/s11042-

020-10139-6.

[24] D. J. Murray-Smith, "Experimental modelling: system identification, parameter estimation and model optimisation techniques", *Model. Simul. Integr. Syst. Eng.*, pp. 165–214, 2012, doi: 10.1533/9780857096050.165.

[25] A. Lambora, K. Gupta, y K. Chopra, "Genetic Algorithm- A Literature Review", *Proc. Int. Conf. Mach. Learn. Big Data, Cloud Parallel Comput. Trends, Prespectives Prospect. Com. 2019*, núm. 1998, pp. 380–384, 2019, doi: 10.1109/COMITCon.2019.8862255.

[26] M. Kumar, M. Husain, N. Upreti, y D. Gupta, "Genetic Algorithm: Review and Application", *SSRN Electron. J.*, vol. 2, núm. 2, pp. 451–454, 2020, doi: 10.2139/ssrn.3529843.

[27] H. Millwater, J. D. Ocampo, y T. Castaldo, "Probabilistic damage tolerance analysis for general aviation", *Adv. Mater. Res.*, vol. 891–892, pp. 1191–1196, 2014, doi: 10.4028/www.scientific.net/AMR.891-892.1191.

[28] P. Kora y P. Yadlapalli, "Crossover Operators in Genetic Algorithms: A Review", *Int. J. Comput. Appl.*, vol. 162, núm. 10, pp. 34–36, 2017, doi: 10.5120/ijca2017913370.

[29] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, y V. B. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach", *Inf.*, vol. 10, núm. 12, 2019, doi: 10.3390/info10120390.

[30] N. Maheswaran, S. P. Venkatesan, M. S. Sampath Kumar, G. Velmurugan, N. Sathishkumar, y M. Priya, "Study of Weight Optimization on Spar Beam for the Wing of an Aircraft", *Int. J. Softw. Hardw. Res. Eng.*, vol. 3, núm. 3, pp. 72–79, 2015, [En línea]. Disponible en: www.ijournals.in

[31] D. A. Skinn, U. of Dayton. Research Institute, A. F. M. L. (U.S.), Metals, y C. I. C. (U.S.), *Damage Tolerant Design Handbook: A Compilation of Fracture and Crack Growth Data for High-strength Alloys*, núm. v. 1. CINDAS/Purdue University, 1994. [En línea]. Disponible en: https://books.google.com.co/books?id=7ng3uAEACAAJ

[32] G.A. Giraldo Echeverri "Análisis estadístico de fractura mecánica y optimización de inspecciones en aeronaves pequeñas", Trabajo de grado profesional, Ingeniería Mecánica, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022

# ANEXO

A continuación, anexo el código desarrollado para el análisis del cronograma una aeronave y de la flota

## *A. Código para cronograma de una aeronave*

```
import random as rd
from random import randint
import numpy as np
import pandas as pd
import time
import subprocess
import pylab as pl
import csv
import matplotlib.pyplot as plt
import time as t
import os


def all_posibilities(n_possibilities,n_times):
    h=(np.zeros((n_possibilities, n_times),dtype=int)).tolist()
    #MATRIX OF ZEROS
    for j in range(0,n_possibilities):
        a=(list(reversed((np.base_repr(j,n_inspeciones)))))
        #COMBINATION OF POSIBILITIES
        for i in range(0,(len(a))):
             h[j][(i)]=int(a[i])
                #INSERTION OF POSIBILITIES IN MATRIX OF ZEROS
    return h

def init_population(n_possibilities,n_times,population_initial,h):
    #SELECTION OF INITIAL POPULATION
    c=(np.random.randint(0, n_possibilities, size=population_initial, dtype=int))
    #RANDOM NUMBERS FOR ELECTION OF INITIAL POPULATION
    init_pop=[]
    for i in range(0,population_initial):
        init_pop.append(h[c[i]])
    # remove individuals duplicates
    init_pop_without_repeated=[]
    for i in init_pop:
        if i not in init_pop_without_repeated:
            init_pop_without_repeated.append(i)
    init_pop=init_pop_without_repeated
    for i in range(0,(len(init_pop))):
        if init_pop[i] ==[0, 0, 0, 0, 0, 0, 0]:
            init_pop.pop(i)
            break

    return init_pop
```

```
def fitness_calculation(init_pop,value,CONTADOR,total_maintenance):
    for i in range (0,len(init_pop)):
        if total_maintenance[i]!=0:
            continue
        else:
            CONTADOR+=1
            total_maintenance[i]=((sum((np.equal(init_pop[i], 0)*0).tolist()))+
                                  (sum(np.equal(init_pop[i], 1)*value[0]).tolist())+
                                  (sum(np.equal(init_pop[i], 2)*value[1]).tolist())+
                                  (sum(np.equal(init_pop[i], 3)*value[2]).tolist()))
            x=np.array(init_pop[i]) #individual
            xs=(x.ravel()[np.flatnonzero(x)]).tolist()
            #remove zeros from the vector of individuals
            insp=' '.join(map(str, xs))
            # string that represents the individual
            times=(np.arange(4000,30000,4000))
            # vector of times
            times=(times.ravel()[np.flatnonzero(x)]).tolist()
            #times of inspection
            times_insp=' '.join(map(str, times))
            #string of times of inspection
            inspection=("INSPECTIONS = "+times_insp+" \n"\
            "INSPECTION_TYPE = "+insp+" ")
            replace=("INSPECTIONS = ")
            #TEXT to replace on SMART
            with open("Capstone.dat","r") as dat_file:
                dat_file2=dat_file.read()
                with open('Capstone_ins.dat','w') as dat_file_ins:
                    dat_file_ins.write(dat_file2.replace(replace,inspection))
                    #REplace
            os.remove("Capstone_ins_amis_data.bin.save")
            #run smart-dt
            program = 'smartdta.exe'
            argument = 'Capstone_ins.dat'

            subprocess.call([program, argument])    #Run Smart-Dt

            with open('Capstone_ins_pof.csv') as csv_file:
                csv_reader = csv.reader(csv_file, delimiter=',')
                tabla=[]
                for row in csv_reader:
                    tabla.append(row)
                x=[]
                y=[]
                for row in range(3,len(tabla)):
                    x.append(float(tabla[row][0]))
                    y.append(float(tabla[row][4]))
                for insrow in range(0,len(y)):
                    if y[insrow] >= 0.0000001:
                        total_maintenance[i]=100
                        break
    return total_maintenance,CONTADOR


def selection(fitness,init_pop):
    list1 = fitness
    list2 = list(init_pop)
    ordered_fitness,prospective_parents=list(zip(*sorted(zip(list1,list2),reverse
    #se ordena la poblacion inicial
    n_parents=int(len(prospective_parents)/3) #truncate selection
    prospective_parents=list(prospective_parents)
    del prospective_parents[n_parents:]
    return prospective_parents,ordered_fitness
```

```
def crossover(parents):
    offspring=[]
    #vector vacio para crear la desencdecia
    n=int(len(parents)-1)      #numero de hijos a crear x 4
    for i in range (0,n):
        child = parents[i][0:3]+parents[i+1][3:6]+parents[i][6:]
        offspring.append(child)

        child = parents[i+1][0:3]+parents[i][3:6]+parents[i+1][6:]
        offspring.append(child)
        child = parents[0][0:3]+parents[i+1][3:6]+parents[0][6:]
        offspring.append(child)

        child = parents[i+1][0:3]+parents[0][3:6]+parents[i+1][6:]
        offspring.append(child)
    offspring_without_repeated=[]
    for i in offspring:
        if i not in offspring_without_repeated:
            offspring_without_repeated.append(i)
    offspring_without_repeated
    init_pop=offspring_without_repeated
    for i in range(0,(len(init_pop))):
        if init_pop[i] ==[0, 0, 0, 0, 0, 0, 0]:
            init_pop.pop(i)
            break
    init_pop=list(map(list, init_pop))
    return init_pop

def mutation(offsprings,n_times,n_inspections):
    number_of_individuals_to_mutate=int(((len(offsprings))*.80)) #10%
    election_of_individuals=(np.random.randint(0,len(offsprings), size=number_of_
    for i in range (0,number_of_individuals_to_mutate) :
        j=rd.randint(0,(n_times-1))
        new_gen=rd.randint(0,n_inspections-1)
        offsprings[election_of_individuals[i]][j]=new_gen
    for i in range(0,(len(offsprings))):
        if offsprings[i] ==[0, 0, 0, 0, 0, 0, 0]:
            offsprings.pop(i)
            break
    return(offsprings)
```

```
n_times=7
n_inspeciones=4 #
n_possibilities=n_inspeciones**(n_times)
population_initial=25
value=[0.1,1,5]
CONTADOR=0
data_pop=[]
data_fitness=[]
v=100
# starting time
start = t.time()
h=all_posibilities(n_possibilities,n_times)
init_pop=init_population(n_possibilities,n_times,population_initial,h)
total_maintenance=[0]*(len(init_pop))
for i in range(0,30):
    total_maintenance,CONTADOR=fitness_calculation(init_pop,value,CONTADOR,total_
    data_pop.extend(init_pop)
    data_fitness.extend(total_maintenance)
    prospective_parents,ordered_fitness=selection(total_maintenance,init_pop)
    solution=prospective_parents[0]
    fitness_v= ordered_fitness[0]
    if v>fitness_v:
        print("iteration {} : best solution={}: costo={}".format(i,solution,fitne
        v=fitness_v # value of reference for the next iteration
    offsprings=crossover(prospective_parents)
    offsprings=mutation(offsprings,n_times,n_inspeciones)
    init_pop=init_population(n_possibilities,n_times,population_initial,h)
    offsprings.extend(init_pop)
    offsprings_without_repeat=[]
    for j in offsprings:
        if j not in offsprings_without_repeat:
            offsprings_without_repeat.append(j)
    offsprings=offsprings_without_repeat
    repeat=[]
    for n in offsprings:
        if n  in data_pop:
            repeat.append(n)
    total_maintenance=[0]*(len(offsprings))
    for _ in range(0,len(offsprings)):
        if repeat is None:
            pass
        else:
            for j in range(0,len(repeat)):
                total_maintenance[offsprings.index(repeat[j])]=data_fitness[data_
    init_pop=offsprings
# end time VBN
end = t.time()
time=end - start
print("total iteration {} : total individuals evaluated {} of {} -total time ={}
```

## B. *Código para una flota*

```
import random as rd
from random import randint
import numpy as np
import pandas as pd
import time
import subprocess
import pylab as pl
import csv
import matplotlib.pyplot as plt
import time as t
import os
from sklearn.metrics import auc

def all_posibilities(n_possibilities,n_times):
    h=(np.zeros((n_possibilities, n_times),dtype=int)).tolist()
    #MATRIX OF ZEROS
    for j in range(0,n_possibilities):
        a=(list(reversed((np.base_repr(j,n_inspeciones)))))
        #COMBINATION OF POSIBILITIES
        for i in range(0,(len(a))):
            h[j][(i)]=int(a[i])
            #INSERTION OF POSIBILITIES IN MATRIX OF ZEROS
    return h
```

```
def init_population(n_possibilities,n_times,population_initial,h):
    #SELECTION OF INITIAL POPULATION
    c=(np.random.randint(0, n_possibilities, size=population_initial, dtype=int))
    #RANDOM NUMBERS FOR ELECTION OF INITIAL POPULATION
    init_pop=[]
    for i in range(0,population_initial):

        init_pop.append(h[c[i]])

    # remove individuals duplicates
    init_pop_without_repeated=[]
    for i in init_pop:
        if i not in init_pop_without_repeated:
            init_pop_without_repeated.append(i)
    init_pop=init_pop_without_repeated
    for i in range(0,(len(init_pop))):
        if init_pop[i] ==[0, 0, 0, 0, 0, 0, 0]:
            init_pop.pop(i)
            break

    return init_pop

def fitness_calculation(init_pop,value,CONTADOR,total_maintenance,POF_DATA):
    for i in range (0,len(init_pop)):
        if total_maintenance[i]!=0:
            continue
        else:
            CONTADOR+=1
            total_maintenance[i]=((sum((np.equal(init_pop[i], 0)*0).tolist()))
                        +(sum(np.equal(init_pop[i], 1)*value[0]).tolist())
                        +(sum(np.equal(init_pop[i], 2)*value[1]).tolist())
                        +(sum(np.equal(init_pop[i], 3)*value[2]).tolist()))

            x=np.array(init_pop[i]) #individual
            xs=(x.ravel()[np.flatnonzero(x)]).tolist()
            #remove zeros from the vector of individuals
            insp=' '.join(map(str, xs))
            # string that represents the individual
            times=(np.arange(1000,8000,1000))
            # vector of times
            times=(times.ravel()[np.flatnonzero(x)]).tolist()
            #times of inspection
            times_insp=' '.join(map(str, times))
            #string of times of inspection
            inspection=("INSPECTIONS = "+times_insp+" \n"\
            "INSPECTION_TYPE = "+insp+" ")
            replace=("INSPECTIONS = ")   #TEXTO A REMPLAZAR EN SMART
            with open("Capstone.dat","r") as dat_file:
                dat_file2=dat_file.read()
                with open('Capstone_ins.dat','w') as dat_file_ins:
                    dat_file_ins.write(dat_file2.replace(replace,inspection))
                    #REMPLAZO
            os.remove("Capstone_ins_amis_data.bin.save")
            #CORRER PROGRAMA SMART
            program = 'smartdta.exe'
            argument = 'Capstone_ins.dat'

            subprocess.call([program, argument])    #Run Smart-Dt

            with open('Capstone_ins_pof.csv') as csv_file:
                csv_reader = csv.reader(csv_file, delimiter=',')
                tabla=[]
                for row in csv_reader:
                    tabla.append(row)
                x=[]
                y=[]
                for row in range(3,len(tabla)):
                    x.append(float(tabla[row][0]))
                    y.append(float(tabla[row][4]))
                index=Index(x,y)
                A_1=auc(x[index[0]:index[1]+1],y[index[0]:index[1]+1])
                A_2= auc(x[index[2]:index[3]+1],y[index[2]:index[3]+1])
                A_3=auc(x[index[4]:index[5]+1],y[index[4]:index[5]+1])
                A_4=auc(x[index[6]:index[7]+1],y[index[6]:index[7]+1])

                POF=A_1+A_2+A_3+A_4
                POF_DATA[i]=POF
                if POF >= 0.0000001:
                    total_maintenance[i]=100

    return total_maintenance,CONTADOR,POF_DATA
```

```
def selection(fitness,init_pop,POF_DATA):
    list1 = fitness
    list2 = list(init_pop)
    ordered_fitness, prospective_parents = list(zip(*sorted(zip(list1, list2),rev
    #se ordena la poblacion inicial
    if ordered_fitness[0]==100:
        list1 = POF_DATA
        list2 = list(init_pop)
        ordered_POF, prospective_parents = list(zip(*sorted(zip(list1, list2),rev
        #se ordena la poblacion inicial
    n_parents=int(len(prospective_parents)/3) #truncate selección
    prospective_parents=list(prospective_parents)
    del prospective_parents[n_parents:]
    return prospective_parents,ordered_fitness


def crossover(parents):
    offspring=[]     #vector vacio para crear la desencdecia
    n=int(len(parents)-1)      #numero de hijos a crear x 4
    for i in range (0,n):
        # 0 up to 4
        child = parents[i][0:3]+parents[i+1][3:6]+parents[i][6:]
        offspring.append(child)

        child = parents[i+1][0:3]+parents[i][3:6]+parents[i+1][6:]
        offspring.append(child)
        child = parents[0][0:3]+parents[i+1][3:6]+parents[0][6:]
        offspring.append(child)

        child = parents[i+1][0:3]+parents[0][3:6]+parents[i+1][6:]
        offspring.append(child)

    offspring_without_repeated=[]
    for i in offspring:
        if i not in offspring_without_repeated:
            offspring_without_repeated.append(i)
    offspring_without_repeated
    init_pop=offspring_without_repeated
    for i in range(0,(len(init_pop))):
        if init_pop[i] ==[0, 0, 0, 0, 0, 0, 0]:
            init_pop.pop(i)
            break
    init_pop=list(map(list, init_pop))
    return init_pop


def mutation(offsprings,n_times,n_inspections):
    number_of_individuals_to_mutate=int(((len(offsprings))*.80)) #10%
    election_of_individuals=(np.random.randint(0,len(offsprings), size=number_of_
    for i in range (0,number_of_individuals_to_mutate) :
        j=rd.randint(0,(n_times-1))
        new_gen=rd.randint(0,n_inspections-1)
        offsprings[election_of_individuals[i]][j]=new_gen
    for i in range(0,(len(offsprings))):
        if offsprings[i] ==[0, 0, 0, 0, 0, 0, 0]:
            offsprings.pop(i)
            break
    return(offsprings)
```

```
def Index(x,y):
    # AIRCRAFT 1
    N_1=1000  #AERONAVE 1-#VUELOS
    N_1=float(N_1)
    I_1=(x.index(N_1))
    E_1=1000   #AERONAVE 1-#VUELOS ESPERADOS
    E_1=float(N_1+E_1)
    I_1_1=(x.index(E_1))
    # AIRCRAFT 2
    N_2=2500  #AERONAVE 2-#VUELOS
    N_2=float(N_2)
    I_2=(x.index(N_2))
    E_2=500   #AERONAVE 2-#VUELOS ESPERADOS
    E_2=float(N_2+E_2)
    I_2_2=(x.index(E_2))
    # AIRCRAFT 3
    N_3=5000  #AERONAVE 3-#VUELOS
    N_3=float(N_3)
    I_3=(x.index(N_3))
    E_3=500   #AERONAVE 3-#VUELOS ESPERADOS
    E_3=float(N_3+E_3)
    I_3_3=(x.index(E_3))
    # AIRCRAFT 4
    N_4=6000  #AERONAVE 4-#VUELOS
    N_4=float(N_4)
    I_4=(x.index(N_4))
    E_4=500   #AERONAVE 4-#VUELOS ESPERADOS
    E_4=float(N_4+E_4)
    I_4_4=(x.index(E_4))
    index=[I_1, I_1_1, I_2, I_2_2,I_3,I_3_3,I_4,I_4_4]
    return (index)
```

```
n_times=7
n_inspeciones=4 #
n_possibilities=n_inspeciones**(n_times)
population_initial=25
value=[0.1,1,5]
CONTADOR=0
data_pop=[]
data_fitness=[]
v=100
# starting time
start = t.time()
h=all_posibilities(n_possibilities,n_times)
init_pop=init_population(n_possibilities,n_times,population_initial,h)
total_maintenance=[0]*(len(init_pop))
POF_DATA=[0]*(len(init_pop))
for i in range(0,30):
    total_maintenance,CONTADOR,POF_DATA=fitness_calculation(init_pop,value,CONTAD
    data_pop.extend(init_pop)
    data_fitness.extend(total_maintenance)
    prospective_parents,ordered_fitness=selection(total_maintenance,init_pop,POF_
    solution=prospective_parents[0]
    fitness_v= ordered_fitness[0]
    if v>=fitness_v:
        print("iteration {} : best solution={}: costo={}".format(i,solution,fitne
        v=fitness_v # value of reference for the next iteration
    offsprings=crossover(prospective_parents)
    offsprings=mutation(offsprings,n_times,n_inspeciones)
    init_pop=init_population(n_possibilities,n_times,population_initial,h)
    offsprings.extend(init_pop)
    offsprings_without_repeat=[]
    for j in offsprings:
        if j not in offsprings_without_repeat:
            offsprings_without_repeat.append(j)
    offsprings=offsprings_without_repeat
    repeat=[]
    for n in offsprings:
        if n  in data_pop:
            repeat.append(n)
    total_maintenance=[0]*(len(offsprings))
    POF_DATA=[0]*(len(offsprings))
    for _ in range(0,len(offsprings)):
        if repeat is None:
            pass
        else:
            for j in range(0,len(repeat)):
                total_maintenance[offsprings.index(repeat[j])]=data_fitness[data_
    init_pop=offsprings
# end time VBN
end = t.time()
time=end - start
print("total iteration {} : total individuals evaluated {} of {} -total time ={} s
```