INFORME PRÁCTICA

Enlace al repositorio de la práctica: PaulaCuesta/Practica3_AndresPaula

Objetivo de la práctica: Implementar un algoritmo genético para optimizar la asignación de turnos del personal de enfermería, asegurando el cumplimiento de restricciones laborales y preferencias individuales

Resumen de las tareas a realizar:

- Cargar Datos: Implementar la carga programática de los datos de enfermería desde el archivo datos_enfermeria.txt.
- Configurar Algoritmo Genético: Definir parámetros y operadores (mutación, cruce, elitismo) para optimizar la calendarización.
- Ejecución y Resultados: Ejecutar el algoritmo con diferentes configuraciones y analizar los resultados obtenidos.
- Estudio de Parámetros: Comparar el impacto de cambios en el número de enfermeros y preferencias.

Para realizar la práctica se han realizado diferentes notebooks:

Turnosenfermería.py: La clase TurnosEnfermeria sirve para evaluar la calidad de las soluciones generadas por el algoritmo genético. Tiene métodos para medir cómo un calendario propuesto cumple o infringe las reglas y preferencias establecidas, ayudando a guiar el algoritmo hacia soluciones más óptimas. (Ya proporcionada)

elitism.py: El algoritmo eaSimpleWithElitism es una variante del algoritmo genético clásico eaSimple de DEAP, que incorpora un mecanismo de elitismo para mejorar la retención de las mejores soluciones durante el proceso evolutivo. Este mecanismo asegura que los individuos de mayor calidad se preserven en cada generación, inyectándolos directamente en la población siguiente sin someterlos a los operadores genéticos de cruce y mutación. (Ya proporcionado)

alg_genetico_simple.py : implementación del código genético. Cargamos los datos, creamos el algoritmo genético, creamos la función plot_evolucion.

- Carga de datos: Carga los datos de preferencias de turnos de los enfermeros desde un archivo CSV.
- Creación del algoritmo genético: Configura y ejecuta un algoritmo genético para generar y evaluar calendarios de turnos.
  Define la estructura de los individuos (calendarios) y la población inicial.

Registra funciones para cruce, mutación, y evaluación del fitness usando las preferencias de los enfermeros.
Ejecuta el algoritmo genético con diferentes estrategias de selección y cruce.

- plot_evolution: Visualiza la evolución de los valores de fitness a través de las generaciones.
  Grafica los valores mínimo, máximo, y promedio de fitness por generación.
  Rellena el área entre los mínimos y máximos para visualizar la dispersión.

EjemplosEjecucion.ipynb: contiene cada una de las situaciones que vamos a explicar a continuación, aplicando el algoritmo genético, que importamos de **alg_genetico_simple.py.**

Situación 1

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de un punto
Tipo de mutación: Mutación Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

Esta configuración es bastante estándar y puede ser efectiva para muchos problemas. La selección por torneo mantiene una buena diversidad en la población. El cruce de un punto es simple, pero puede ser menos efectivo que otras variantes para problemas complejos. La mutación Flip-Bit es adecuada para problemas de optimización binaria.

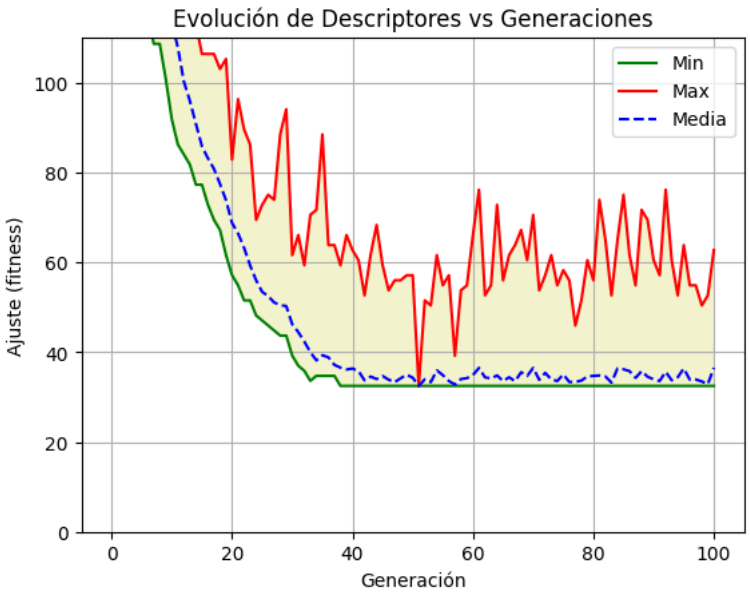
Situación 2

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de dos puntos
Tipo de mutación: Mutación Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

Esta configuración es similar a la Situación 1, pero utiliza cruce de dos puntos, que generalmente es más efectivo que el cruce de un punto, ya que permite un intercambio más variado de información genética.

```
Calendario de cada enfermer@:
Enf_A : T M T L M M M
Enf_B : M T L L M L M
Enf_C : L M M M L N L
Enf_D : M T T L T M T
Enf_E : N T M L M L N
Enf_F : L L T M D N M
Enf_G : L T L M M L N
Enf_H : L L T M N T L
Enf_I : T M M T L M L
Enf_J : M M L L L M T
Enf_K : T M L T N L M
Enf_L : N N L T T M T
Enf_M : M L L N L M T

Infracciones por turnos consecutivos =  13


Infracciones por turnos el mismo día =  1

Turnos por semana =  [7, 5, 4, 8, 5, 6, 5, 5, 8, 5, 5, 7, 5]
Infracciones de turnos por semana =  11

Enfermer@ por turno =  [4, 5, 2, 5, 6, 2, 3, 4, 1, 4, 4, 2, 5, 3, 3, 6, 4, 2, 4, 4, 2]
Infracciones por enfermer@ por turno =  4

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

[7]:
```
1 alg_gen.plot_evolucion(logbook2)
```

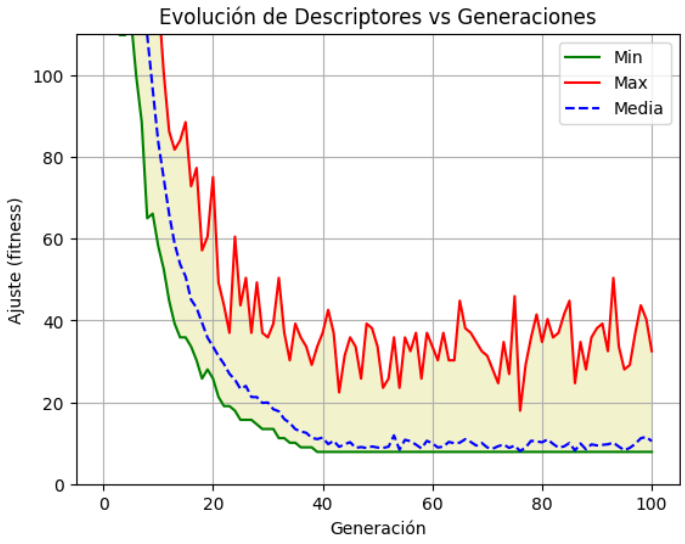


Situación 3

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce uniforme
Tipo de mutación: Mutación Flip-Bit
Tipo de algoritmo: Algoritmo genético simpleEl cruce uniforme puede ser más efectivo que el cruce de uno o dos puntos en ciertos tipos de problemas, especialmente aquellos con alta epistasis. Esta configuración podría ser más robusta que las dos anteriores.

```
Calendario de cada enfermer@:
Enf_A : T M L T M T L
Enf_B : T M L T M M L
Enf_C : N T L L T N N
Enf_D : T N T L M M L
Enf_E : N L M N L M M
Enf_F : L L M M N L M
Enf_G : M T M L L N T
Enf_H : L L L L T T M
Enf_I : M L N L T M M
Enf_J : M L L M L T T
Enf_K : L M M M L L T
Enf_L : M N T T L L T
Enf_M : L M T M M L L

Infracciones por turnos consecutivos =  2


Infracciones por turnos el mismo día =  0

Turnos por semana =  [6, 5, 5, 5, 5, 4, 5, 3, 5, 4, 4, 5, 5]
Infracciones de turnos por semana =  1

Enfermer@ por turno =  [4, 3, 2, 4, 3, 2, 4, 3, 1, 4, 4, 1, 4, 3, 1, 4, 3, 2, 4, 4, 1]
Infracciones por enfermer@ por turno =  4

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [9]:  1  alg_gen.plot_evolucion(logbook3)
```



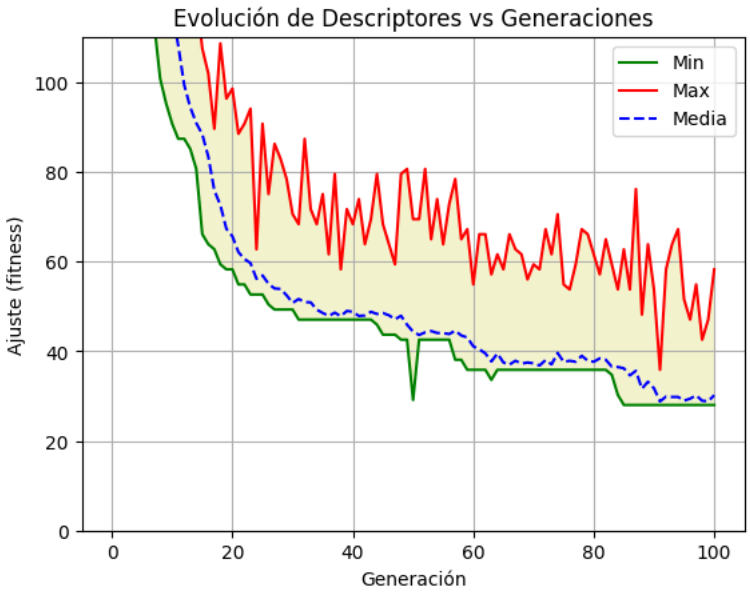
Situación 4

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce en representación de orden
Tipo de mutación: Mutación Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

El cruce en representación de orden es más adecuado para problemas de permutación, como el problema del viajante. Sin embargo, la mutación Flip-Bit no es la más apropiada para este tipo de representación.

```
Calendario de cada enfermer@:
Enf_A : N L L M L T M
Enf_B : L T N L N L M
Enf_C : M M L M T N L
Enf_D : N L M T L L N
Enf_E : L L M M M L L
Enf_F : M M T N D L L
Enf_G : L T L L L M M
Enf_H : L M T M M N T
Enf_I : M T M T L M L
Enf_J : T M L M N T D
Enf_K : M M L L T L T
Enf_L : L N M N M M M
Enf_M : L M T M L N L

Infracciones por turnos consecutivos =  10


Infracciones por turnos el mismo día =  2

Turnos por semana =  [4, 5, 5, 4, 5, 6, 5, 6, 5, 7, 6, 6, 4]
Infracciones de turnos por semana =  6

Enfermer@ por turno =  [4, 2, 2, 6, 3, 2, 4, 4, 1, 6, 2, 2, 4, 3, 4, 3, 3, 3, 5, 3, 2]
Infracciones por enfermer@ por turno =  7

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
n [11]:  1  alg_gen.plot_evolucion(logbook4)
```



Situación 5

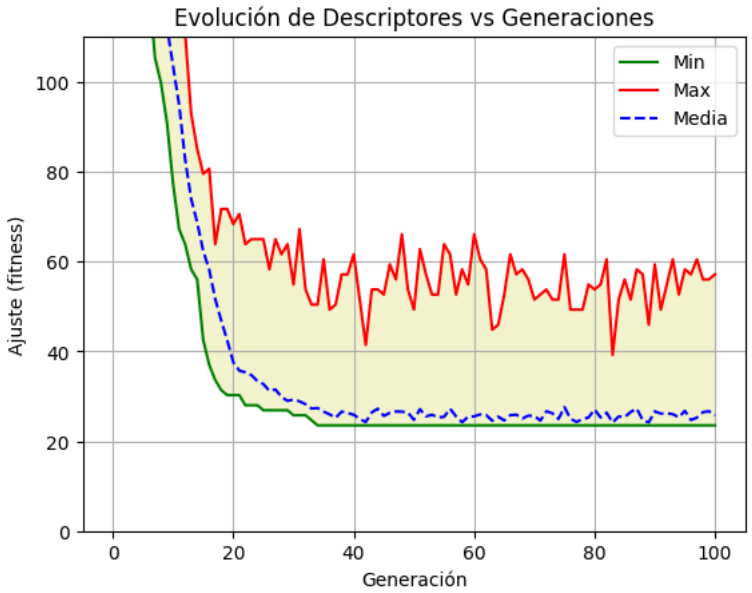
Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de un punto

Tipo de mutación: Mutación de Mezcla de Índices (Shuffle)
Tipo de algoritmo: Algoritmo genético simple

La mutación de Mezcla de Índices es más apropiada para problemas de permutación. Esta configuración podría ser efectiva para problemas de optimización combinatoria.

```
Calendario de cada enfermer@:
Enf_A : L L L N L M M
Enf_B : N T M T M D L
Enf_C : L T N T M L L
Enf_D : M M L M L T L
Enf_E : L T N T L L N
Enf_F : L D M M M L L
Enf_G : L L L L T T M
Enf_H : L N T L L L T
Enf_I : M T T N L L L
Enf_J : L L T M N L M
Enf_K : M L M L L T L
Enf_L : M M M L T L N
Enf_M : M L L L T M M

Infracciones por turnos consecutivos =  8


Infracciones por turnos el mismo día =  2

Turnos por semana =  [3, 7, 4, 5, 4, 6, 3, 3, 5, 5, 4, 6, 5]
Infracciones de turnos por semana =  4

Enfermer@ por turno =  [5, 3, 1, 3, 4, 2, 4, 4, 2, 3, 4, 2, 3, 3, 1, 3, 3, 2, 4, 2, 2]
Infracciones por enfermer@ por turno =  7

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
13]:  1  alg_gen.plot_evolucion(logbook5)
```



Situación 6

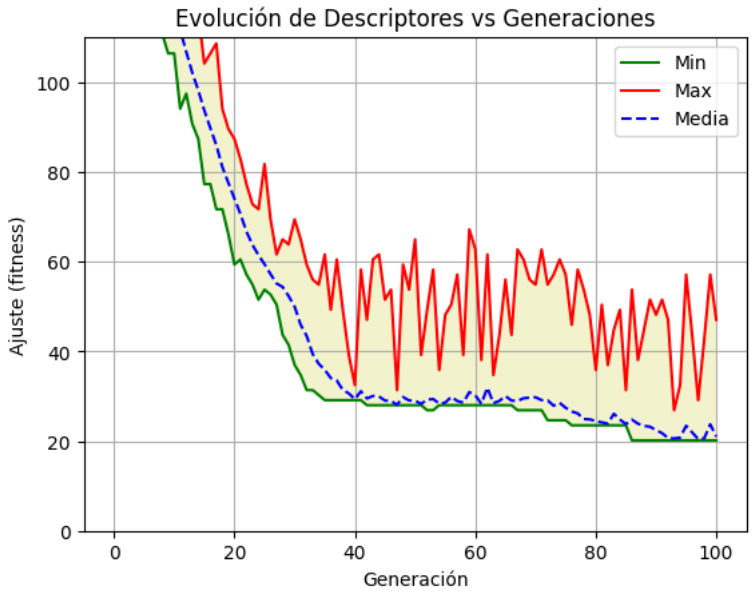
Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de un punto

Tipo de mutación: Mutación de Inversión
Tipo de algoritmo: Algoritmo genético simple

La mutación de Inversión también es adecuada para problemas de permutación. Esta configuración podría ser efectiva para problemas como el del viajante.

```
Calendario de cada enfermer@:
Enf_A : T N L T D N N
Enf_B : M M L T T T L
Enf_C : M M N M M L M
Enf_D : T L L M T T T
Enf_E : L T L M M L N
Enf_F : L M T L L M T
Enf_G : L T T M T L M
Enf_H : M T M M L M T
Enf_I : L L M T T T M
Enf_J : M D L T L N L
Enf_K : T M M N L M T
Enf_L : N L N N M M M
Enf_M : N L T L N L T

Infracciones por turnos consecutivos =  5


Infracciones por turnos el mismo día =  2

Turnos por semana =  [7, 7, 6, 5, 4, 4, 5, 6, 5, 5, 6, 7, 4]
Infracciones de turnos por semana =  9

Enfermer@ por turno =  [4, 3, 2, 5, 4, 2, 3, 3, 2, 5, 4, 2, 4, 5, 2, 4, 3, 3, 4, 5, 2]
Infracciones por enfermer@ por turno =  2

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
n [15]:  1 alg_gen.plot_evolucion(logbook6)
```



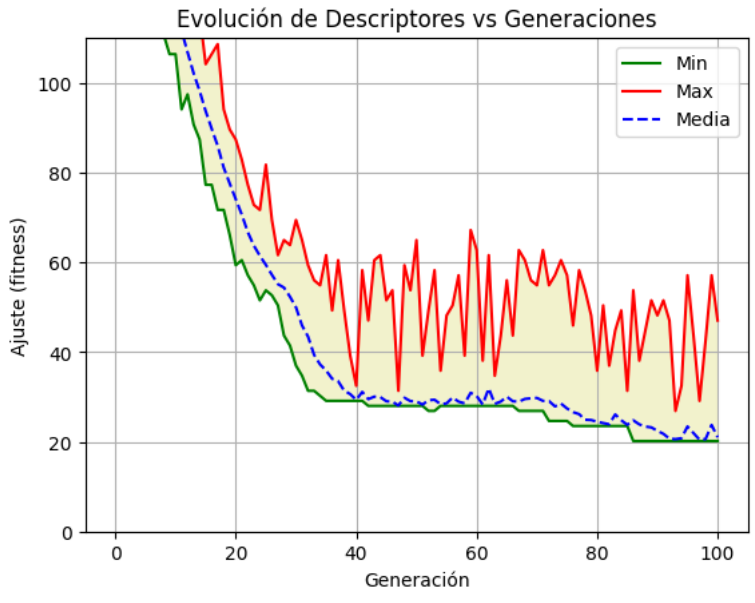
Situación 7

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de dos puntos
Tipo de mutación: Mutación de mezcla de Índices
Tipo de algoritmo: Algoritmo genético simple

Esta configuración combina el cruce de dos puntos con la mutación de mezcla de índices, lo que podría ser efectivo para una variedad de problemas de optimización combinatoria.

```
Calendario de cada enfermer@:
Enf_A : M T L L M M L
Enf_B : L N M D L T L
Enf_C : L L L T N T L
Enf_D : N T M L M M L
Enf_E : L N L L T T L
Enf_F : T L M T L N L
Enf_G : L M N L L M M
Enf_H : T L N M M L L
Enf_I : M M T L T L T
Enf_J : L M L M N M T
Enf_K : N L M L M L M
Enf_L : L M L T T L M
Enf_M : T T T L L L L

Infracciones por turnos consecutivos =  9


Infracciones por turnos el mismo día =  1

Turnos por semana =  [5, 5, 4, 5, 3, 5, 4, 4, 5, 5, 5, 5, 4]
Infracciones de turnos por semana =  0

Enfermer@ por turno =  [2, 4, 2, 4, 3, 3, 4, 3, 2, 3, 3, 1, 4, 3, 2, 4, 3, 2, 3, 4, 0]
Infracciones por enfermer@ por turno =  8

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [18]:  1  alg_gen.plot_evolucion(logbook6)
```



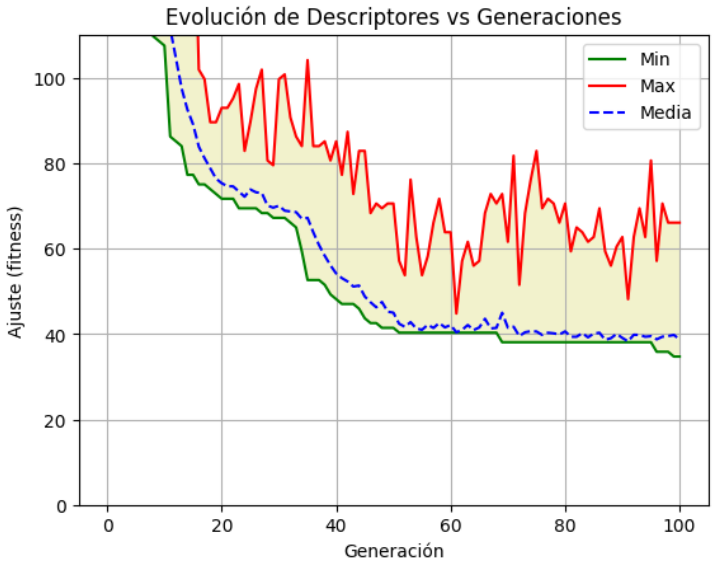
Situación 8

Tipo de selección: Selección por torneo, con tamaño = 3
Tipo de cruce: Cruce de un punto

Tipo de mutación: Mutación de Flip Bit
Tipo de algoritmo: Algoritmo simple con Elitismo

La introducción del elitismo puede mejorar la convergencia y la calidad de la solución final, especialmente en problemas donde se busca maximizar el fitness.

```
Calendario de cada enfermer@:
Enf_A : L T M L D L L
Enf_B : M T L M T L L
Enf_C : T M L L T N M
Enf_D : L T M L M L M
Enf_E : T M T D T L L
Enf_F : T T N N L M M
Enf_G : L L M T M L T
Enf_H : M L N M L L N
Enf_I : M T D L M N T
Enf_J : M D L T L T L
Enf_K : M M T M D T L
Enf_L : L L T D L M T
Enf_M : N N N L L M T

Infracciones por turnos consecutivos =  9


Infracciones por turnos el mismo día =  6

Turnos por semana =  [5, 5, 5, 5, 6, 7, 5, 4, 8, 5, 7, 6, 5]
Infracciones de turnos por semana =  9

Enfermer@ por turno =  [5, 4, 1, 4, 5, 2, 4, 4, 4, 5, 3, 3, 5, 5, 2, 3, 3, 2, 3, 5, 1]
Infracciones por enfermer@ por turno =  7

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```



Situación 9

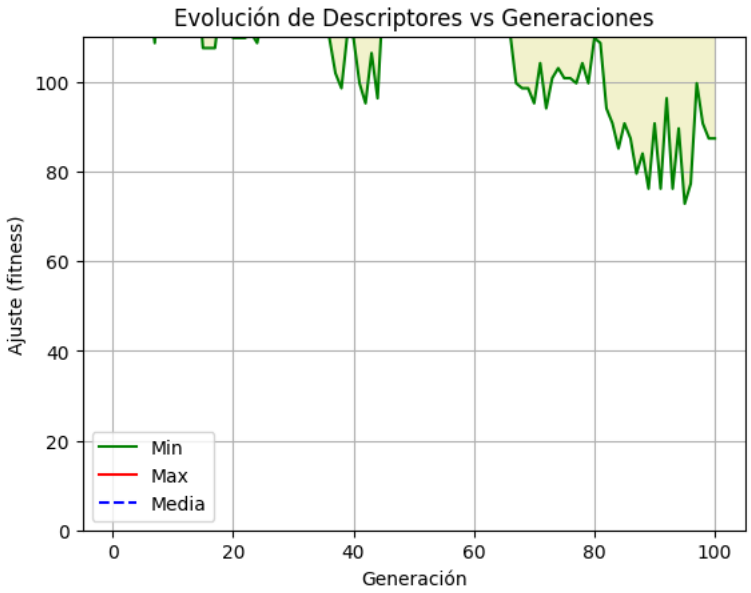
Tipo de selección: Selección de los K-mejores
Tipo de cruce: Cruce de un punto
Tipo de mutación: Mutación de Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

Esta configuración utiliza la selección de los K-mejores, que es más elitista que la selección por torneo. Esto puede llevar a una convergencia más rápida, pero también

puede reducir la diversidad de la población. El cruce de un punto y la mutación Flip-Bit son opciones estándar.

```
Calendario de cada enfermer@:
Enf_A : L L T L N D N
Enf_B : L M M D T L L
Enf_C : M L M T N T L
Enf_D : N M L M M L L
Enf_E : T T M M M M M
Enf_F : T T L T M N L
Enf_G : T T L M L L T
Enf_H : L L D L D T M
Enf_I : M L D L N M N
Enf_J : N N T L L N L
Enf_K : M T M M N L M
Enf_L : N L L M T D T
Enf_M : L L D L M M T

Infracciones por turnos consecutivos =  22


Infracciones por turnos el mismo día =  7

Turnos por semana =  [5, 6, 6, 6, 8, 7, 5, 8, 7, 5, 8, 7, 5]
Infracciones de turnos por semana =  18

Enfermer@ por turno =  [3, 4, 4, 2, 5, 2, 7, 4, 4, 6, 4, 2, 5, 2, 5, 5, 3, 6, 3, 4, 3]
Infracciones por enfermer@ por turno =  18

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [22]:  1  alg_gen.plot_evolucion(logbook9)
```



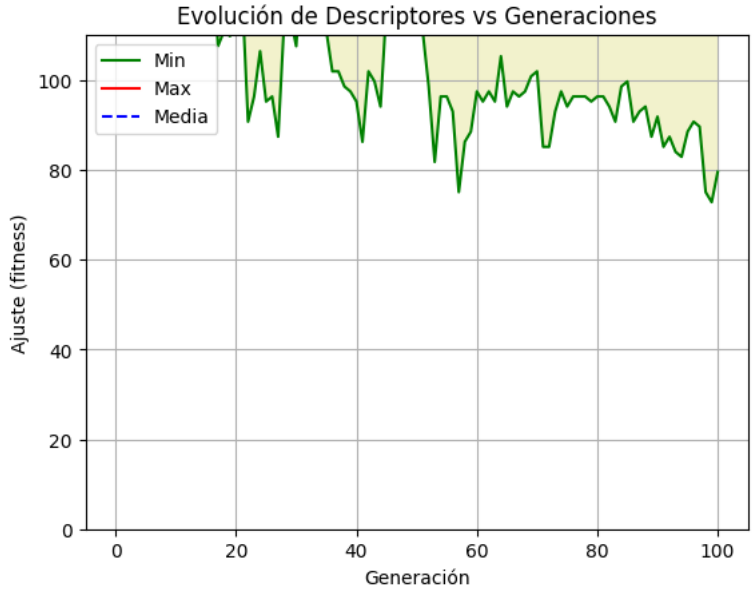
Situación 10

Tipo de selección: Selección de los K-mejores
Tipo de cruce: Cruce de dos puntos
Tipo de mutación: Mutación de Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

Similar a la situación 9, pero utiliza cruce de dos puntos, que generalmente es más efectivo que el cruce de un punto para mantener la diversidad genética.

```
Calendario de cada enfermer@:
Enf_A : M M L L M N L
Enf_B : M N L L M T T
Enf_C : T L L N M N T
Enf_D : M D T M T T L
Enf_E : T T M L N M D
Enf_F : M M M L L M D
Enf_G : T L M N L T L
Enf_H : T L T L M L L
Enf_I : L D L L N M M
Enf_J : D L M M T D T
Enf_K : L T T L D M M
Enf_L : L M T N L L L
Enf_M : D L N T T M T

Infracciones por turnos consecutivos =  22


Infracciones por turnos el mismo día =  8

Turnos por semana =  [5, 5, 5, 10, 8, 8, 4, 3, 6, 11, 9, 3, 8]
Infracciones de turnos por semana =  25

Enfermer@ por turno =  [6, 7, 2, 5, 4, 3, 4, 5, 2, 2, 3, 3, 5, 4, 4, 6, 5, 3, 4, 5, 3]
Infracciones por enfermer@ por turno =  10

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [24]:  1  alg_gen.plot_evolucion(logbook10)
```



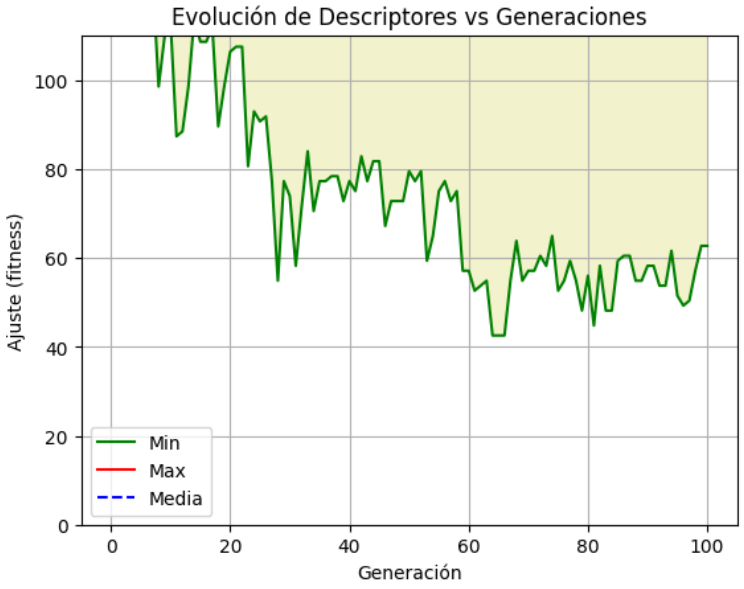
Situación 11

Tipo de selección: Selección de los K-mejores
Tipo de cruce: Cruce uniforme
Tipo de mutación: Mutación de Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

El cruce uniforme puede ser más efectivo que el cruce de uno o dos puntos en ciertos tipos de problemas, especialmente aquellos con alta epistasis. Esta configuración podría ser más robusta que las dos anteriores.

```
Calendario de cada enfermer@:
Enf_A : D M D N M L T
Enf_B : M L L T L M L
Enf_C : M D T M L L T
Enf_D : L T M L M T L
Enf_E : L T L T M L L
Enf_F : L L L L L M L
Enf_G : T T L L L N D
Enf_H : L L N T T L T
Enf_I : M N L L M L L
Enf_J : M L L M T L L
Enf_K : M L T N T T L
Enf_L : L M T L M L M
Enf_M : T L T L L L N

Infracciones por turnos consecutivos =  11


Infracciones por turnos el mismo día =  4

Turnos por semana =  [9, 3, 6, 4, 4, 1, 6, 6, 4, 4, 5, 5, 3]
Infracciones de turnos por semana =  7

Enfermer@ por turno =  [6, 3, 2, 3, 3, 2, 2, 4, 2, 2, 3, 3, 5, 6, 0, 2, 2, 1, 2, 4, 3]
Infracciones por enfermer@ por turno =  16

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [26]:  1  alg_gen.plot_evolucion(logbook11)
```



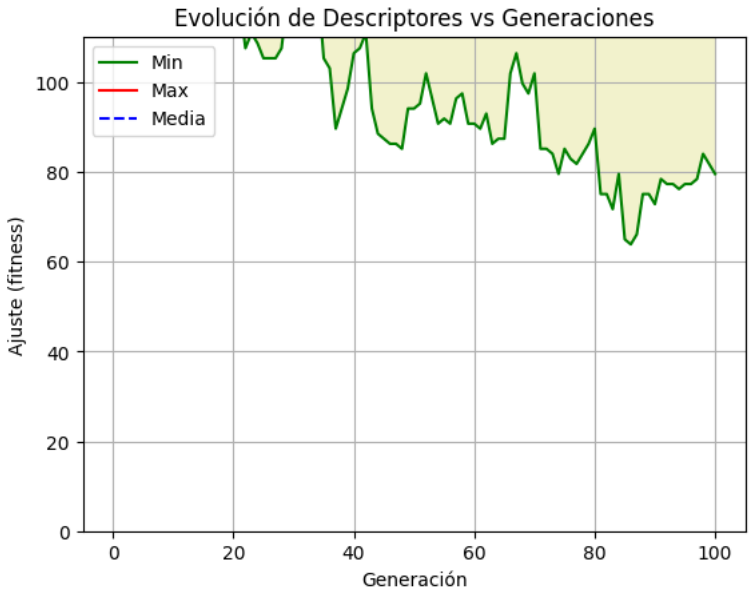
Situación 12

Tipo de selección: Selección de los K-mejores
Tipo de cruce: Cruce en representación de Orden
Tipo de mutación: Mutación de Flip-Bit
Tipo de algoritmo: Algoritmo genético simple

El cruce en representación de orden es más adecuado para problemas de permutación. Sin embargo, la mutación Flip-Bit no es la más apropiada para este tipo de representación, lo que podría limitar la efectividad de esta configuración.

```
Calendario de cada enfermer@:
Enf_A : T N N L N T L
Enf_B : L L D L L T L
Enf_C : N M N L T M M
Enf_D : L M L M M N T
Enf_E : L L L L L N M
Enf_F : T L L D M L L
Enf_G : M T M L D N L
Enf_H : N N T M T M M
Enf_I : T D L L L L L
Enf_J : L M T L M M M
Enf_K : L T D M D T N
Enf_L : D L T D T M T
Enf_M : M N N N L L L

Infracciones por turnos consecutivos =  16


Infracciones por turnos el mismo día =  8

Turnos por semana =  [6, 3, 6, 6, 2, 5, 6, 7, 5, 6, 9, 9, 4]
Infracciones de turnos por semana =  15

Enfermer@ por turno =  [3, 3, 5, 4, 4, 4, 3, 3, 5, 5, 0, 3, 5, 5, 4, 4, 3, 3, 4, 2, 2]
Infracciones por enfermer@ por turno =  18

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [28]:  1  alg_gen.plot_evolucion(logbook12)
```



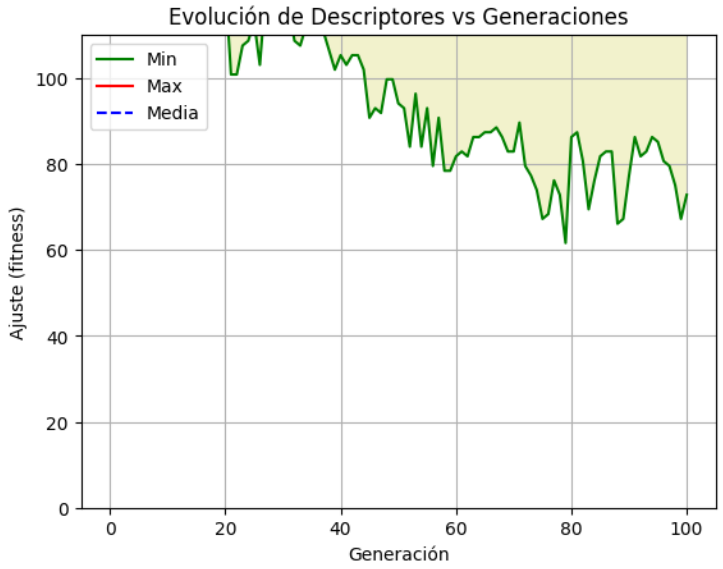
Situación 13

Tipo de selección: Selección de los K-Mejores
Tipo de cruce: Cruce de un punto
Tipo de mutación: Mutación de mezcla de Índices
Tipo de algoritmo: Algoritmo genético simple

La mutación de mezcla de índices es más apropiada para problemas de permutación. Esta configuración podría ser efectiva para problemas de optimización combinatoria.

```
Calendario de cada enfermer@:
Enf_A : M T L M L L L
Enf_B : N L D L D T D
Enf_C : M N N N N M D
Enf_D : L T M N N L T
Enf_E : L L M L M D T
Enf_F : M M L D L T L
Enf_G : M M L T L T M
Enf_H : L T L M T N M
Enf_I : T T T L T N L
Enf_J : N L T T L N M
Enf_K : M T L M M L T
Enf_L : T M T T L L D
Enf_M : M L M T M T N

Infracciones por turnos consecutivos =  16


Infracciones por turnos el mismo día =  7

Turnos por semana =  [3, 9, 8, 5, 7, 6, 7, 7, 5, 5, 8, 6, 7]
Infracciones de turnos por semana =  20

Enfermer@ por turno =  [6, 5, 2, 3, 6, 2, 4, 3, 2, 4, 5, 3, 4, 4, 3, 2, 4, 4, 6, 5, 6]
Infracciones por enfermer@ por turno =  12

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [30]:  1  alg_gen.plot_evolucion(logbook13)
```



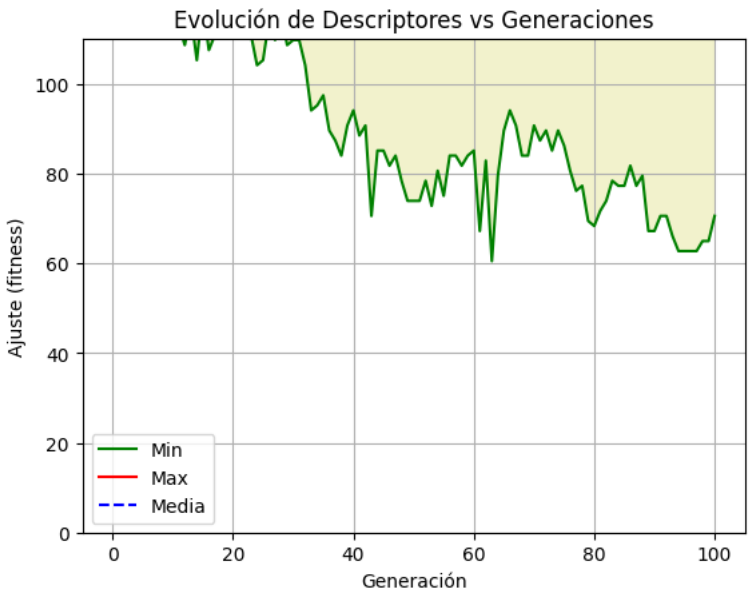
Situación 14

Tipo de selección: Selección de los K-Mejores
Tipo de cruce: Cruce de un punto
Tipo de mutación: Mutación de Inversión
Tipo de algoritmo: Algoritmo genético simple

La mutación de inversión también es adecuada para problemas de permutación. Esta configuración podría ser efectiva para problemas como el del viajante.

```
Calendario de cada enfermer@:
Enf_A : M L N M M L T
Enf_B : M L M N T T M
Enf_C : N L L M M T T
Enf_D : L L T L N M T
Enf_E : L L L D M M M
Enf_F : L D L D D L D
Enf_G : N T M M L T T
Enf_H : M D M T D N L
Enf_I : M N N N T M L
Enf_J : L M T L N L T
Enf_K : L M L N L N M
Enf_L : L M T T M M L
Enf_M : T L D M M T N

Infracciones por turnos consecutivos =  16


Infracciones por turnos el mismo día =  8

Turnos por semana =  [5, 8, 7, 4, 5, 8, 6, 10, 7, 5, 4, 6, 7]
Infracciones de turnos por semana =  19

Enfermer@ por turno =  [4, 4, 2, 5, 2, 3, 4, 5, 3, 6, 3, 5, 7, 2, 4, 4, 5, 2, 4, 5, 3]
Infracciones por enfermer@ por turno =  11

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

[32]:
```
1 alg_gen.plot_evolucion(logbook14)
```



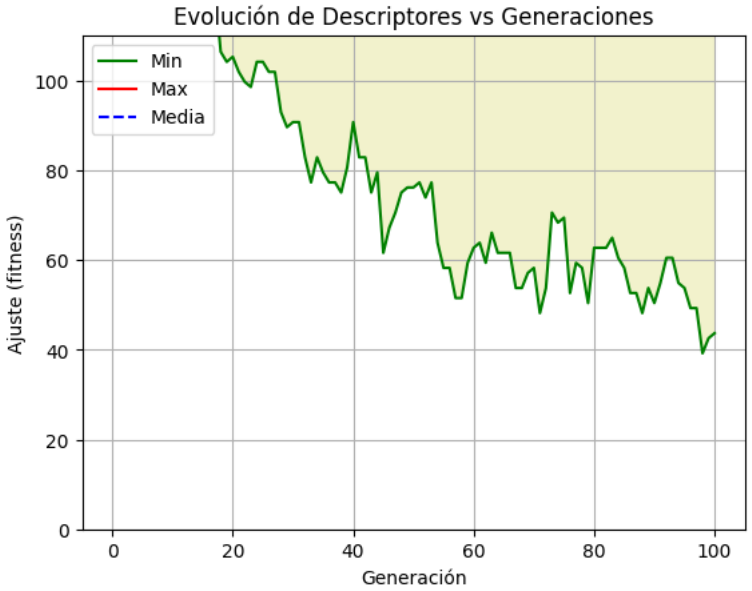
Situación 15

Tipo de selección: Selección de los K-Mejores
Tipo de cruce: Cruce de dos puntos
Tipo de mutación: Mutación de mezcla de índices
Tipo de algoritmo: Algoritmo genético simple

Esta configuración combina el cruce de dos puntos con la mutación de mezcla de índices, lo que podría ser efectivo para una variedad de problemas de optimización combinatoria.

```
Calendario de cada enfermer@:
Enf_A : M T T L N T N
Enf_B : L M T L L M T
Enf_C : L N T T L M N
Enf_D : N L T T T L T
Enf_E : T L L M L L L
Enf_F : L N L M L L N
Enf_G : M L L N L M N
Enf_H : T N L M M L D
Enf_I : M L L T L L M
Enf_J : N M M L L N M
Enf_K : N M M L L L L
Enf_L : L L T M M L L
Enf_M : M M L L L L M

Infracciones por turnos consecutivos =  12


Infracciones por turnos el mismo día =  1

Turnos por semana =  [6, 6, 5, 6, 3, 4, 4, 6, 6, 5, 3, 3, 4]
Infracciones de turnos por semana =  5

Enfermer@ por turno =  [4, 4, 3, 4, 2, 3, 2, 5, 1, 4, 5, 2, 2, 1, 1, 3, 2, 1, 4, 3, 5]
Infracciones por enfermer@ por turno =  17

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [35]:  1  alg_gen.plot_evolucion(logbook15)
```

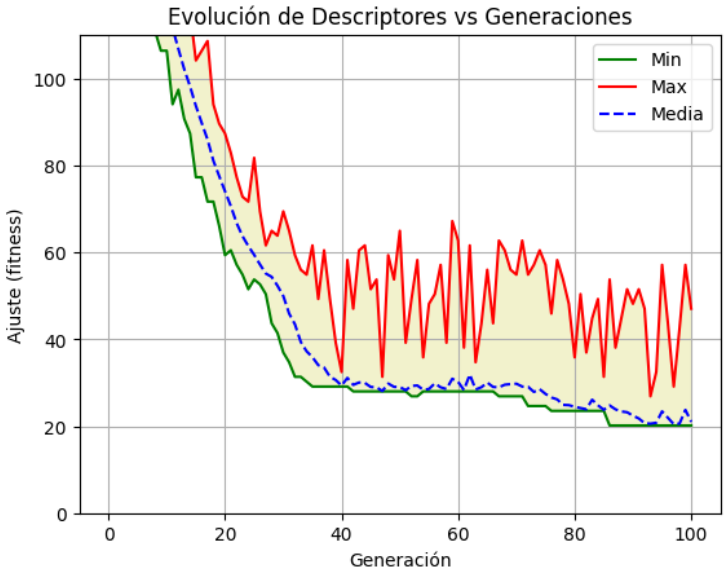


Situación 16

Tipo de selección: Selección de los K-Mejores
Tipo de cruce: Cruce de un punto
Tipo de mutación: Mutación de Flip-Bit
Tipo de algoritmo: Algoritmo genético simple con Elitismo

```
Calendario de cada enfermer@:
Enf_A : M M L L L N N
Enf_B : M M L L M M L
Enf_C : L L N L N L N
Enf_D : T L T N M L M
Enf_E : L N L M T L N
Enf_F : L M T M T T T
Enf_G : T M L L T T N
Enf_H : L N L L M T L
Enf_I : T T N L M L M
Enf_J : L L L M M M M
Enf_K : T L L N L T M
Enf_L : L L T M L N L
Enf_M : D L L L L M T

Infracciones por turnos consecutivos =  6


Infracciones por turnos el mismo día =  1

Turnos por semana =  [4, 5, 3, 5, 5, 6, 5, 3, 6, 5, 4, 4, 4]
Infracciones de turnos por semana =  2

Enfermer@ por turno =  [3, 4, 1, 4, 2, 2, 0, 3, 2, 4, 2, 2, 5, 4, 1, 3, 4, 2, 4, 3, 4]
Infracciones por enfermer@ por turno =  12

Preferencias no cumplidas:  [0, 0, 0, 0]
Infracciones por preferencias de turnos =  0
```

```
In [37]:  1  alg_gen.plot_evolucion(logbook6)
```



Viendo los resultados la mejor opción es la situación 3, Tipo de selección: Selección por torneo, con tamaño = 3, tipo de cruce: Cruce uniforme, tipo de mutación: Mutación Flip-Bit, tipo de algoritmo: Algoritmo genético simple, que tiene 2 fallos por turnos seguidos, no tiene fallos de más de 5 turnos y cumple las preferencias

PREGUNTA 1: ¿Podríamos disponer de un calendario con las mismas restricciones, pero con un enfermer@ menos?

Para saber si es posible tener un calendario que tenga las mismas restricciones, pero con un trabajador menos, lo primero que debemos hacer es analizar como afecta esta falta de un enfermer@ a las restricciones planteadas en el enunciado de la práctica.

- Restricciones por turno: El turno de mañana exige un mínimo de 4 enfermeros disponibles, el turno de tarde un mínimo de 3 enfermeros y el turno de noche un total de 2 enfermeros. Por lo tanto, el que contásemos con un enfermero menos en el servicio de urgencias dificultaría cumplir estas restricciones.

- Restricciones de no trabajar dos turnos seguidos y máximo semanal: Como tenemos un enfermero menos, es mucho más difícil cumplir dicha restricción, puesto que tenemos menos flexibilidad con los horarios del resto de los enfermeros del servicio, además, hay que tener en cuenta la libranza semanal, es decir, cada enfermero solo puede trabajar un máximo de 5 turnos a la semana.

- Preferencias personal: Esta restricción es mucho más difícil de cumplir, puesto que disminuyen las posibilidades de variación de los turnos de los enfermeros. Sin embargo, se trata de una restricción blanda, por lo que es más fácil no cumplirla, o no trae tantas consecuencias como no respetar una restricción dura.

Por lo tanto, teniendo en cuenta las restricciones de nuestro algoritmo, no es posible disponer de un calendario con las mismas restricciones, pero con un trabajador menos, puesto que no contamos con suficientes turnos como para cubrir la totalidad de turnos necesarios.

La única opción viable sería relajar las restricciones del algoritmo, lo cual si que se puede hacer con las preferencias individuales del equipo del servicio, pero no con las duras, puesto que en dicho caso habría repercusiones legales.

- **PREGUNTA 2: ¿Mejoramos el rendimiento del calendario si incluimos un enfermer@ más?**

Si incluyemos un enfermero más en el calendario de turnos mejoraría de forma significativa el rendimiento del calendario, puesto que como tenemos un enfermero más en el servicio, sería mucho más fácil cumplir tanto las restricciones duras como blandas.

Esto conllevaría un menor número de consecuencias legales y una mayor satisfacción entre el personal de urgencias.

Pero lo más importante de todo es que al cumplir con las restricciones establecidas, el coste total se minimizaría, ya que este depende de las infracciones que tengan lugar en el servicio, por lo tanto, conseguiríamos una minimización de nuestro problema, que es lo que buscamos con la función de ajuste (fitness) de minimización que hemos incluido en nuestro algoritmo genético.

Por lo tanto, podemos concluir que mejoraría el rendimiento del calendario con un enfermero más trabajando en el servicio de urgencias.

- **PREGUNTA 3: Imaginemos que aumentamos las preferencias de los enfermer@s (es decir, que hay más enfermer@s que tienen preferencias y los que tienen, aumentan las suyas), ¿sigue siendo válida la configuración elegida?**

Si aumentasen bien el número de preferencias de los enfermeros o el número de enfermeros con preferencias la configuración elegida podría seguir siendo válida, pero su efectividad se verá comprometida por cómo se realiza la gestión por parte del algoritmo y como gestiona el sistema de optimización dichas preferencias.

A su vez, el número incrementado de preferencias aumenta la complejidad del algoritmo genético, ya que es más difícil encontrar un calendario óptimo que respete todas las preferencias de los empleados del servicio. A su vez, esta complejidad aumentada, puede hacer que se incurra en un mayor número de infracciones, debido a que es más difícil respetar las preferencias de los trabajadores.

Por lo tanto, el coste de no cumplir las preferencias debería ajustarse y disminuirse un poco, porque si le damos prioridad al cumplimiento de las preferencias, que son las restricciones blandas del problema, vamos a incurrir en un mayor número de infracciones de las restricciones duras, las cuales son las que más nos interesan respetar.