Anexo 1 – Datos de Entrada

```python
import random

# Datos proporcionados
horarios = {
    'H001': 'L-Mi-V 09:00 - 10:00',
    'H002': 'L-Mi-V 10:00 - 11:00',
    'H003': 'Ma-J 09:00 - 10:30',
    'H004': 'Ma-J 10:30 - 12:00'
}

aulas = {
    'A001': 45,
    'A002': 35,
    'A003': 25
}

profesores = {
    'P001': {'nombre': 'Dr. Edwin Villanueva', 'preferido': None},
    'P002': {'nombre': 'Mg. Layla Hirsh', 'preferido': 'H001'},
    'P003': {'nombre': 'Dr. Manuel Tupia', 'preferido': None},
    'P004': {'nombre': 'Mg. Cesar Aguilera', 'preferido': 'H002'}
}

cursos = {
    'C001': {'nombre': 'Fundamentos de programación', 'alumnos': 45, 'profesores': ['P001', 'P002', 'P003', 'P004']},
    'C002': {'nombre': 'Bases de Datos', 'alumnos': 45, 'profesores': ['P004']},
    'C003': {'nombre': 'Algoritmia', 'alumnos': 35, 'profesores': ['P002', 'P003']},
    'C004': {'nombre': 'Sistemas de información', 'alumnos': 30, 'profesores': ['P003', 'P004']},
    'C005': {'nombre': 'Sistemas de Información 2', 'alumnos': 30, 'profesores': ['P003', 'P004']},
    'C006': {'nombre': 'Machine Learning', 'alumnos': 25, 'profesores': ['P001', 'P002']},
    'C007': {'nombre': 'Deep Learning', 'alumnos': 20, 'profesores': ['P001']}
}
```

FUENTE: Código Python de Solución de Problema

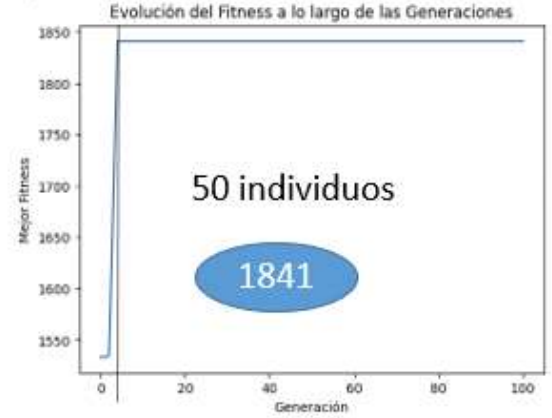
Anexo 2 - Determinación de número de individuos en la población y el número de generaciones

```
Fitness total final para este individuo: 1377
Generación 100, Mejor fitness = 1377
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004',
```


```
Fitness total final para este individuo: 1529
Generación 100, Mejor fitness = 1529
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```


```
Generación 100, Mejor fitness = 1685
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001'
```


```
Fitness total final para este individuo: 1841
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001
```



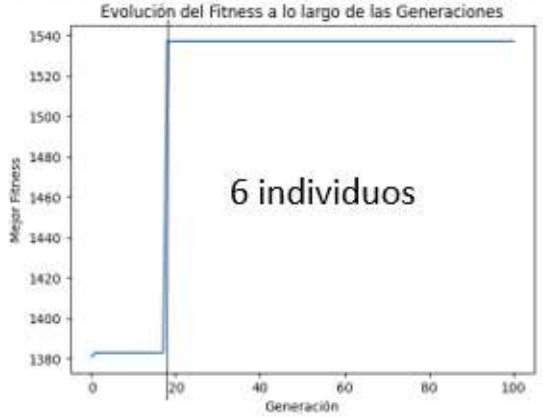

```python
# Hiperparámetros del algoritmo genético
tamano_poblacion = 2 # número de individuos
GENERATIONS = 100  # número de generaciones
PMUT = 0.0  # tasa de mutación

# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="uniform", mutation="multiflip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')
```

FUENTE: Código Python de Solución de Problema

Anexo 3 - Determinación de número de individuos en la población y el número de generaciones

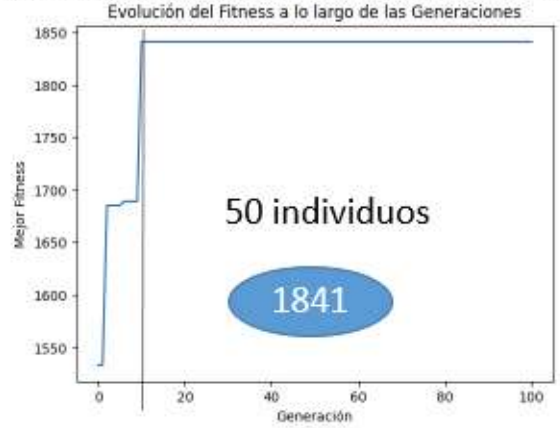
```
Fitness total final para este individuo: 1225
Generación 100, Mejor fitness = 1225
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H002',
```


```
Fitness total final para este individuo: 1537
Generación 100, Mejor fitness = 1537
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'
```




```
Fitness total final para este individuo: 1841
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004
```


```
Fitness total final para este individuo: 1841
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003',
```



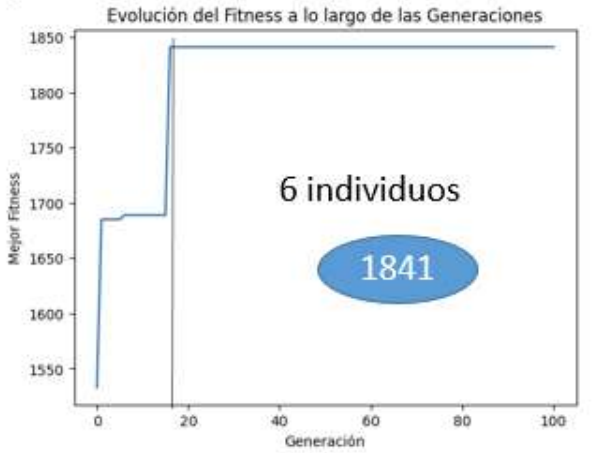

```python
# Hiperparámetros del algoritmo genético
tamano_poblacion = 2  # número de individuos
GENERATIONS = 100  # número de generaciones
PMUT = 0.0  # tasa de mutación

# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="onepoint", mutation="multiflip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')
```

FUENTE: Código Python de Solución de Problema

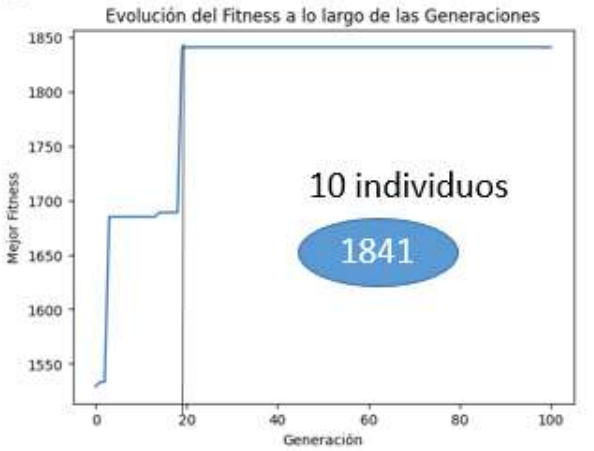
Anexo 4 - Determinación de número de individuos en la población y el número de generaciones

```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```

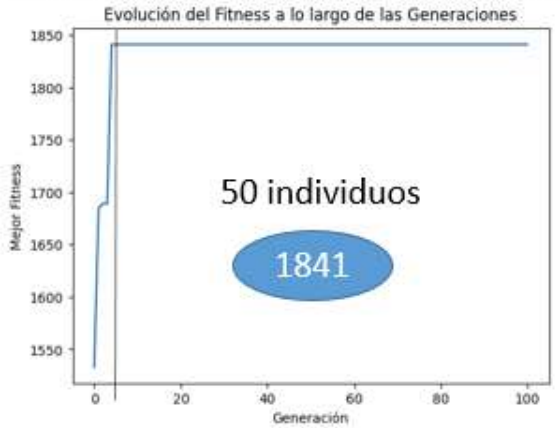

```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```




```
Fitness total final para este individuo: 1841
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003
```




```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```

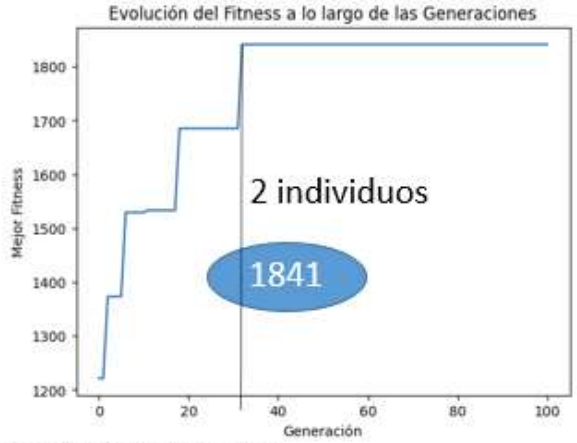



```python
# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="onepoint", mutation="flip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')
```

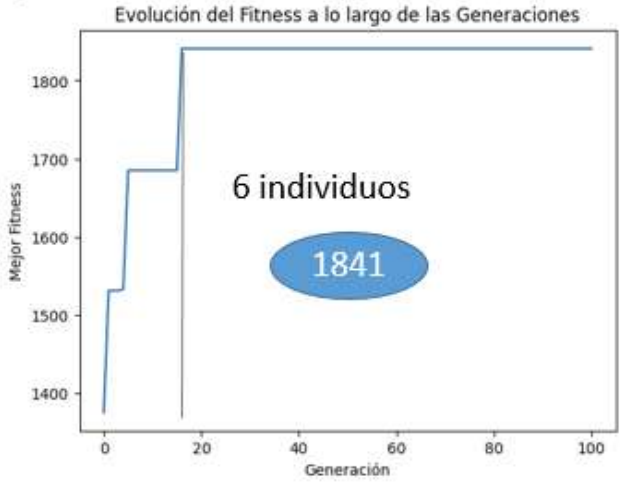
FUENTE: Código Python de Solución de Problema

Anexo 5 - Determinación de número de individuos en la población y el número de generaciones

```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```

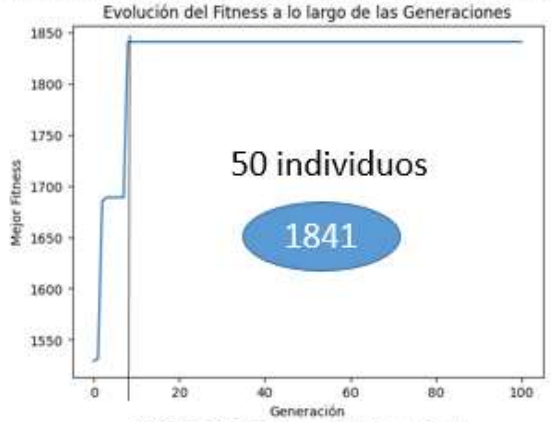



```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001'
```




```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003
```


```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004
```



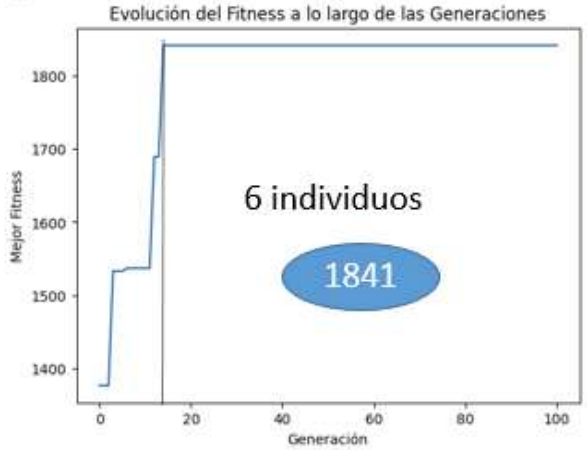

```python
# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="onepoint", mutation="multiflip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')
```

FUENTE: Código Python de Solución de Problema

Anexo 6 - Determinación de número de individuos en la población y el número de generaciones

```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003'
```


```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'
```




```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003
```


```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003
```


```python
# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="uniform", mutation="flip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')

```

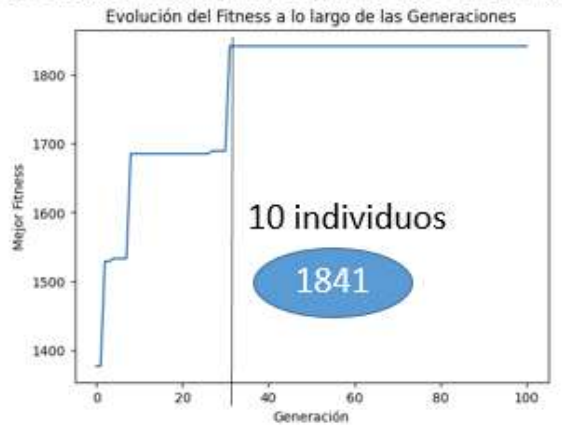
FUENTE: Código Python de Solución de Problema

Anexo 7 - Determinación de número de individuos en la población y el número de generaciones

```
Generación 100, Mejor fitness = 1689
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```


```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H002'
```


```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004
```




```
Generación 100, Mejor fitness = 1841
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00
```

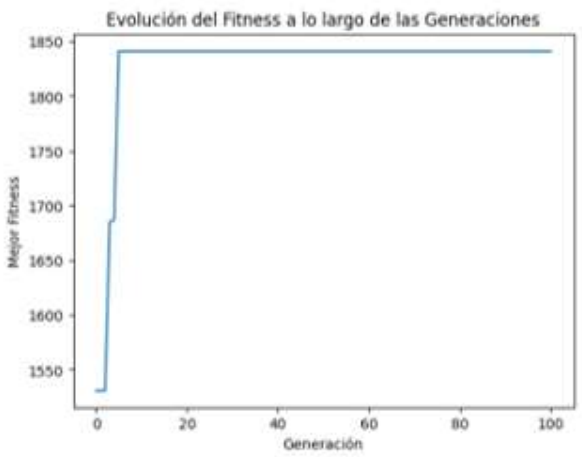

```python
# Hiperparámetros del algoritmo genético
tamano_poblacion = 6 # número de individuos
GENERATIONS = 100  # número de generaciones
PMUT = 0.5  # tasa de mutación

# Inicializa una población inicial de forma aleatoria
poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)

# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                          crossover="uniform", mutation="multiflip",
                                          selection_parents_method='roulette',
                                          selection_survivors_method='ranking')
```

FUENTE: Código Python de Solución de Problema

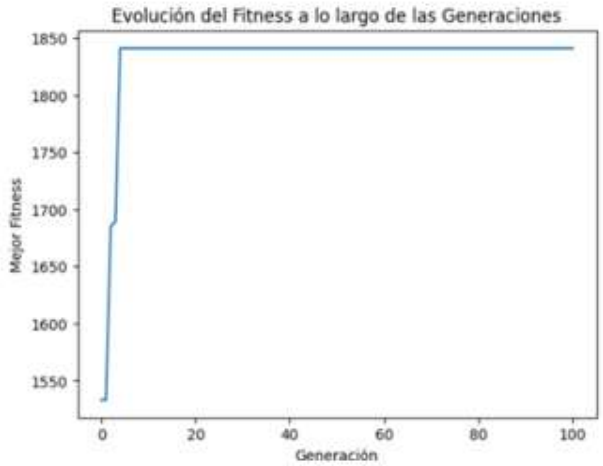
Anexo 8 - Determinación de tipo de mutación

MULTIFLIP , 50 INDIVIDUOS , 100 GENERACIONES

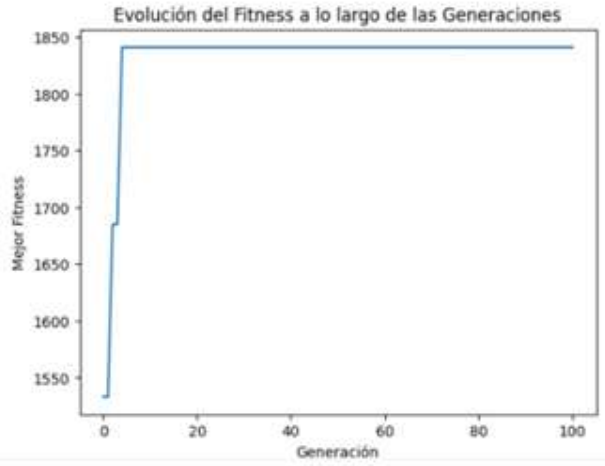
Pmut: 5%



Pmut: 10%



Pmut: 100%

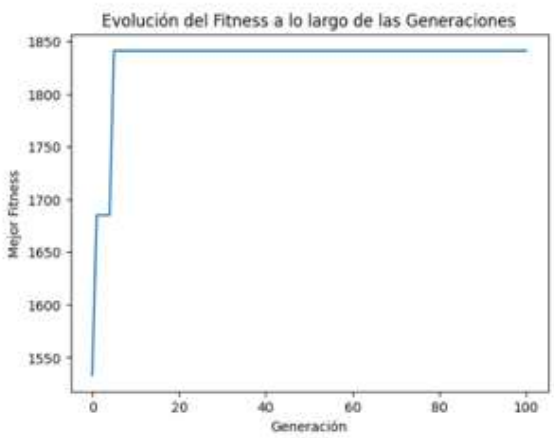


FUENTE: Código Python de Solución de Problema

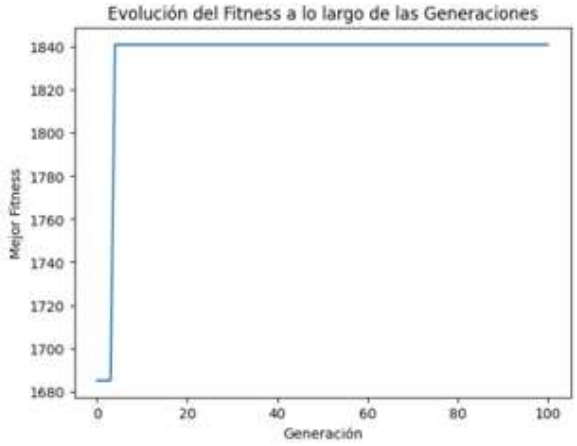
Anexo 9 - Determinación de tipo de mutación

FLIP , 50 INDIVIDUOS , 100 GENERACIONES

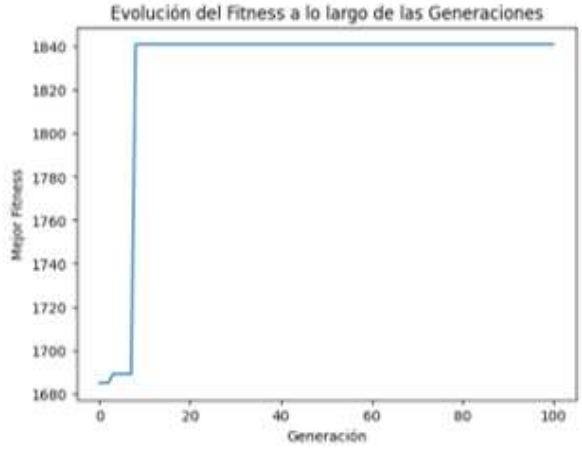
Pmut: 5%



Pmut: 10%

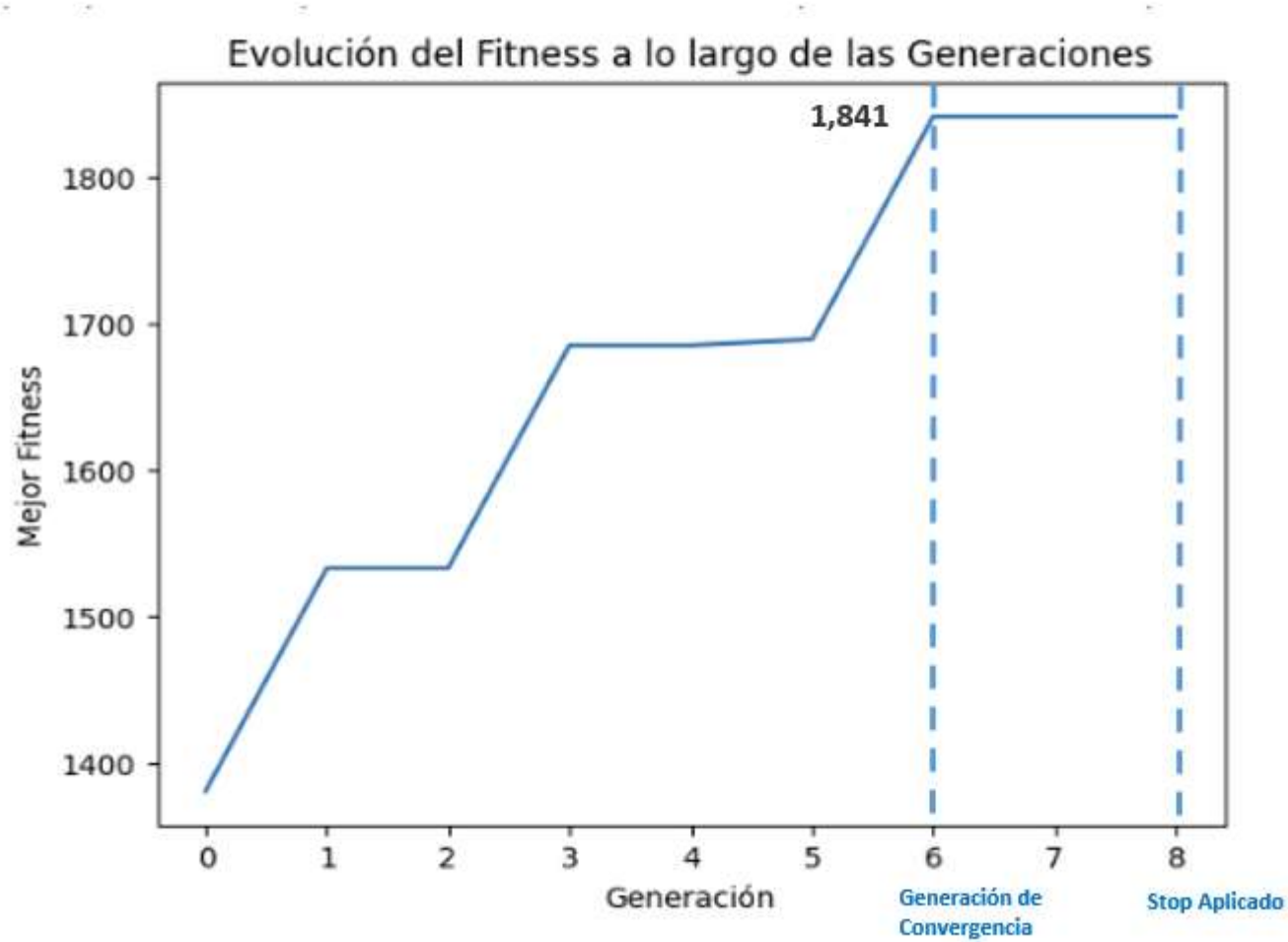


Pmut: 100%



FUENTE: Código Python de Solución de Problema

Anexo 10 Resultados de la Experimentación (Etapa 2)



FUENTE: Código Python de Solución de Problema

Anexo 11 Resultados de la Experimentación (Etapa 2)

1. Output del Código donde se muestra el individuo con mejor fitness y cumpliendo las restricción del caso

```
Fitness de 1841 repetido en las últimas 3 generaciones. Deteniendo.
Mejor individuo en la última generación =
{'curso': 'C001', 'horario': 'H002', 'aula': 'A001', 'profesor': 'P002'}
{'curso': 'C002', 'horario': 'H003', 'aula': 'A001', 'profesor': 'P004'}
{'curso': 'C003', 'horario': 'H004', 'aula': 'A001', 'profesor': 'P002'}
{'curso': 'C004', 'horario': 'H001', 'aula': 'A001', 'profesor': 'P004'}
{'curso': 'C005', 'horario': 'H002', 'aula': 'A002', 'profesor': 'P003'}
{'curso': 'C006', 'horario': 'H003', 'aula': 'A002', 'profesor': 'P001'}
{'curso': 'C007', 'horario': 'H002', 'aula': 'A003', 'profesor': 'P001'}
(fitness = 1841)
```

FUENTE: Código Python de Solución de Problema

2. Output del Código para el usuario final (formato amigable)

	clase_id	curso	cantidad_alumnos	aula_id	capacidad	profesor_id	profesor	horario_id	horario
0	C001	Fundamentos de programación	45	A001	45	P002	Mg. Layla Hirsh	H002	L-Mi-V 10:00 - 11:00
1	C002	Bases de Datos	45	A001	45	P004	Mg. Cesar Aguilera	H003	Ma-J 09:00 - 10:30
2	C003	Algoritmia	35	A001	45	P002	Mg. Layla Hirsh	H004	Ma-J 10:30 - 12:00
3	C004	Sistemas de información	30	A001	45	P004	Mg. Cesar Aguilera	H001	L-Mi-V 09:00 - 10:00
4	C005	Sistemas de Información 2	30	A002	35	P003	Dr. Manuel Tupia	H002	L-Mi-V 10:00 - 11:00
5	C006	Machine Learning	25	A002	35	P001	Dr. Edwin Villanueva	H003	Ma-J 09:00 - 10:30
6	C007	Deep Learning	20	A003	25	P001	Dr. Edwin Villanueva	H002	L-Mi-V 10:00 - 11:00

FUENTE: Código Python de Solución de Problema