

# Aplicación de Algoritmos Genéticos a la optimización de programación de Horarios de clase

## Equipo

- Sandra Atencio
- Piero Tejada
- Manuel Azpilcueta
- Danilo Sanchez

## 1 Descripción del Problema

Elaborar un **Cronograma Semanal para 7 cursos**, asignándole a cada uno:

- *horario*
- *aula*
- *profesor*

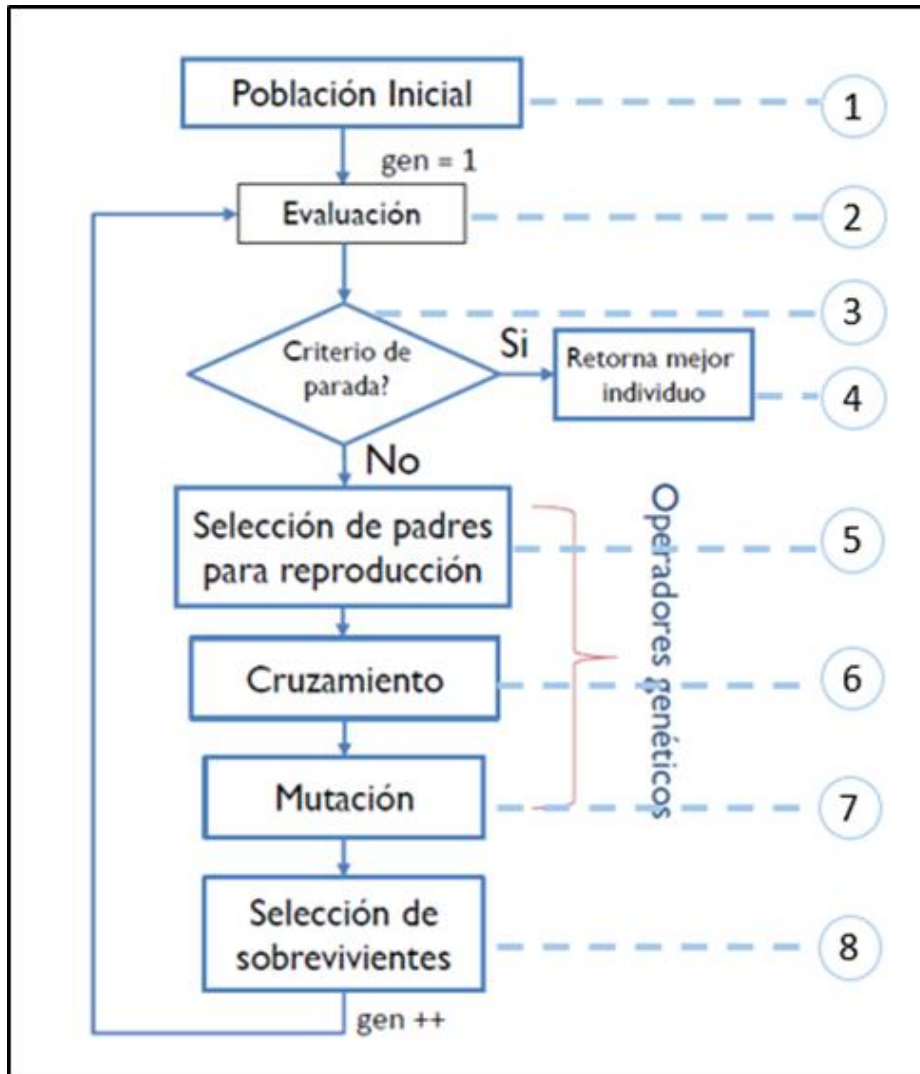
Restricciones obligatorias y opcionales

Anexo 1 – Detalles de Data

## 2 Hipótesis del problema

- Algoritmo Genético
- Evolución de posibles vía mutaciones y cruzamientos
- Determinación y Uso de parámetros adecuados
- **Mejor cronograma semanal** tomando como objetivo el valor de Fitness = **1,841** (= 7 cursos \* 263 puntos máximos por asignaciones correctas por curso).

### 3 Metodología de trabajo adecuada al problema



FUENTE: Metodología presentada en dictado de curso

#### 1. Población Inicial:

Inicia con una población aleatoria de 50 individuos

#### 2. Evaluación:

Evalúa los Fitness de los individuos y obtiene el Fitness máximo

#### 3. Criterio de parada:

Se ha definido como criterio de parada “Número máximo de generaciones”, en caso que el Fitness llegue a 1841 deberá pasar 2 generaciones más para que el sistema pare.

#### 4. Retorno mejor individuo

En caso de encontrar el mejor Fitness= 1841 devuelve valores y finaliza sistema

#### 5. Selección de padres para reproducción:

De forma aleatoria se ha definido la selección de padres por el método “ruleta”

#### 6. Cruzamiento:

Se experimentó Onepoint y Uniform, y en este caso se obtuvo un mejor Fitness con un rápido resultado con “Onepoint” (Ver anexo)

#### 7. Mutación:

De la misma forma que en el punto 6 se obtuvo como resultado el tipo de mutación= ‘MultiFlip’ con una tasa de mutación= 5% (ver anexo)

#### 8. Selección de sobrevivientes:

Se elige a los sobrevivientes mediante el método de Ranking, en este caso a los 50 primeros fitness máximos.

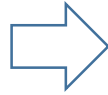
#### Métricas de Evaluación:

1. Fitness de cada individuo
2. Valor de Generación de Convergencia

## 4 Experimentación

### Etapa 1 del Experimento

Determinación de los parámetros óptimos del algoritmo (anexo 2 – 9)

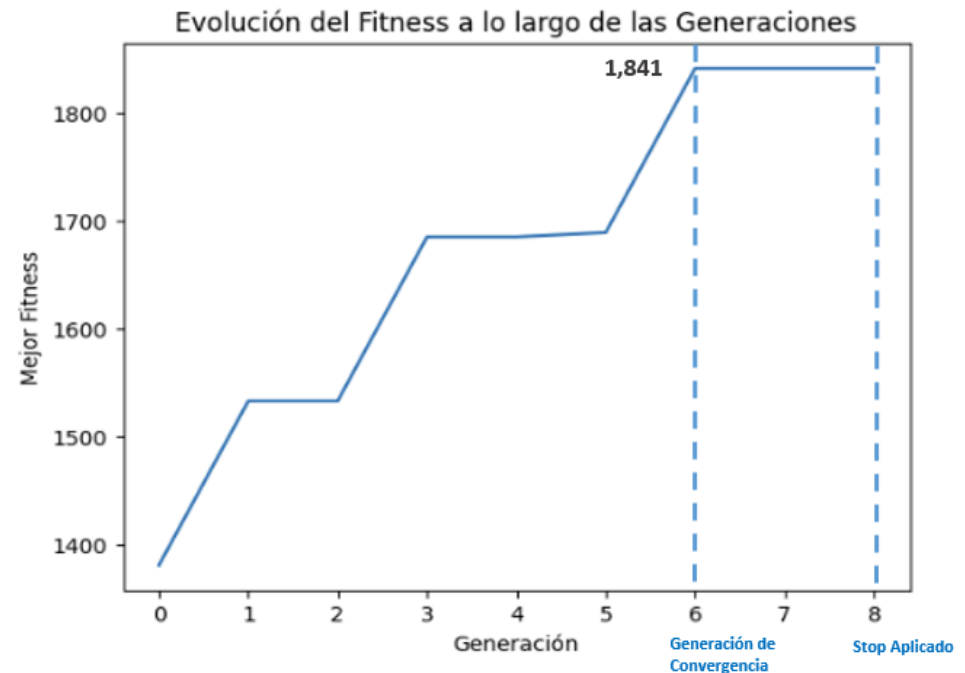
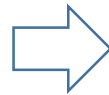


- **Cruzamiento:** One Point
- **Mutación:** MultiFlip
- **Población:** 50
- **Generaciones:** 100
- **Pmut:** 5%



### Etapa 2 del Experimento

Aplicación de los parámetros - previamente definidos - en el algoritmo genético para hallar la mejor solución posible a través del Fitness y el resultado de generaciones requeridas para alcanzar dicho valor.



FUENTE: Resultado de múltiples simulaciones usando código Python de Solución de Problema (Anexo 2-9)



# 4 Resultado

## 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

El enfoque vía algoritmos genéticos demostró ser una **solución efectiva y eficiente** para el problema de asignación de horarios, aulas y profesores a cada uno de los 7 cursos.

- ✓ Eficiencia del Enfoque
- ✓ Optimización de Parámetros
- ✓ Cumplimiento de Restricciones
- ✓ Eficiencia en el Uso de Recursos

**ANEXOS**



## Anexo 1 – Datos de Entrada

```
1 import random
2
3 # Datos proporcionados
4 horarios = {
5     'H001': 'L-Mi-V 09:00 - 10:00',
6     'H002': 'L-Mi-V 10:00 - 11:00',
7     'H003': 'Ma-J 09:00 - 10:30',
8     'H004': 'Ma-J 10:30 - 12:00'
9 }
10
11 aulas = {
12     'A001': 45,
13     'A002': 35,
14     'A003': 25
15 }
16
17 profesores = {
18     'P001': {'nombre': 'Dr. Edwin Villanueva', 'preferido': None},
19     'P002': {'nombre': 'Mg. Layla Hirsh', 'preferido': 'H001'},
20     'P003': {'nombre': 'Dr. Manuel Tupia', 'preferido': None},
21     'P004': {'nombre': 'Mg. Cesar Aguilera', 'preferido': 'H002'}
22 }
23
24 cursos = {
25     'C001': {'nombre': 'Fundamentos de programación', 'alumnos': 45, 'profesores': ['P001', 'P002', 'P003', 'P004']},
26     'C002': {'nombre': 'Bases de Datos', 'alumnos': 45, 'profesores': ['P004']},
27     'C003': {'nombre': 'Algoritmia', 'alumnos': 35, 'profesores': ['P002', 'P003']},
28     'C004': {'nombre': 'Sistemas de información', 'alumnos': 30, 'profesores': ['P003', 'P004']},
29     'C005': {'nombre': 'Sistemas de Información 2', 'alumnos': 30, 'profesores': ['P003', 'P004']},
30     'C006': {'nombre': 'Machine Learning', 'alumnos': 25, 'profesores': ['P001', 'P002']},
31     'C007': {'nombre': 'Deep Learning', 'alumnos': 20, 'profesores': ['P001']}
32 }
33 FUENTE: Código Python de Solución de Problema
```

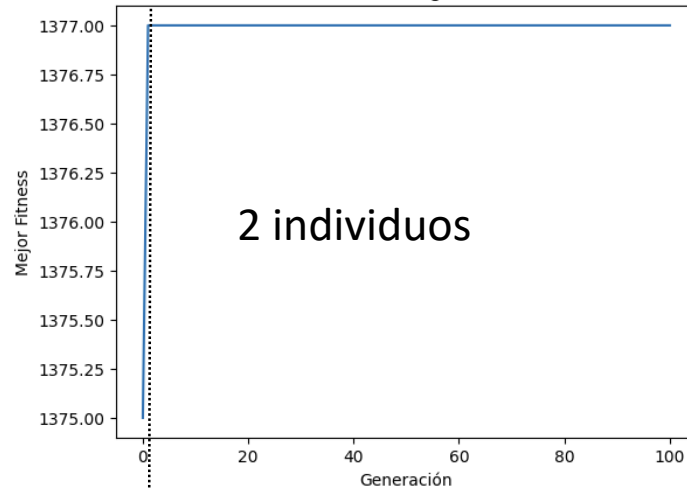
## Anexo 2 - Determinación de número de individuos en la población y el número de generaciones

Fitness total final para este individuo: 1377

Generación 100, Mejor fitness = 1377

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'}]

Evolución del Fitness a lo largo de las Generaciones

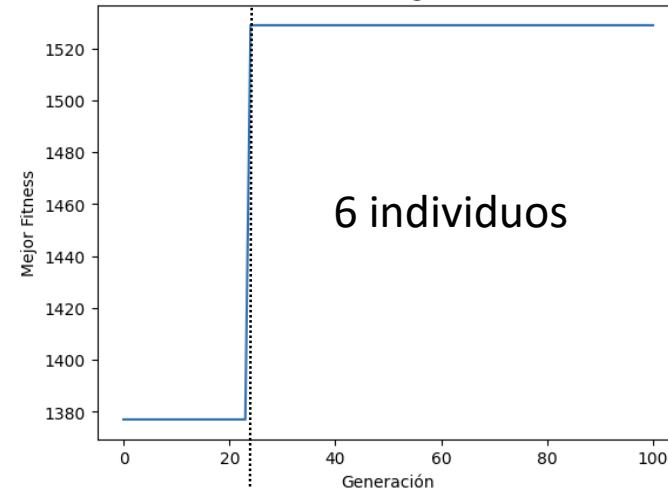


Fitness total final para este individuo: 1529

Generación 100, Mejor fitness = 1529

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'}]

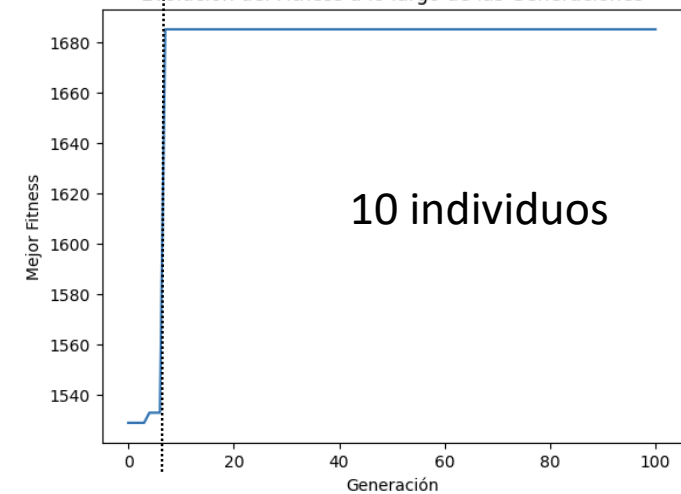
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1685

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001'}]

Evolución del Fitness a lo largo de las Generaciones

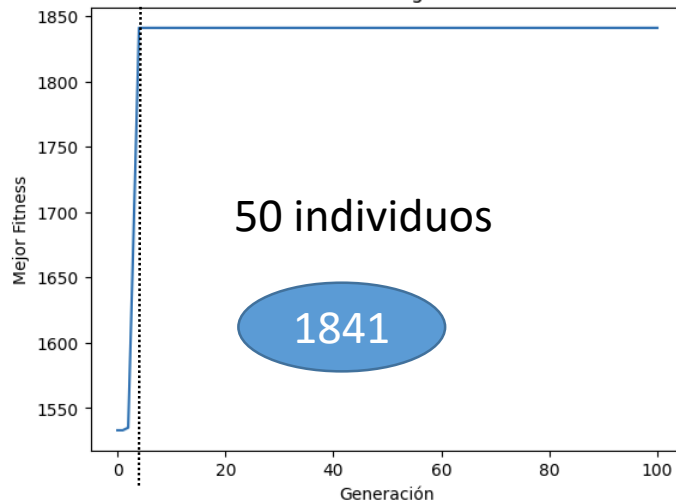


Fitness total final para este individuo: 1841

Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001'}]

Evolución del Fitness a lo largo de las Generaciones



```
# 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)

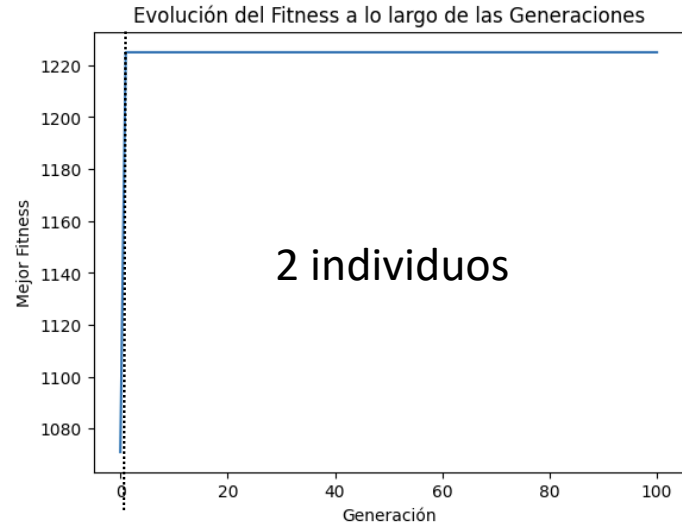
# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                         crossover="uniform", mutation="multiflip",
                                         selection_parents_method='roulette',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

## 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

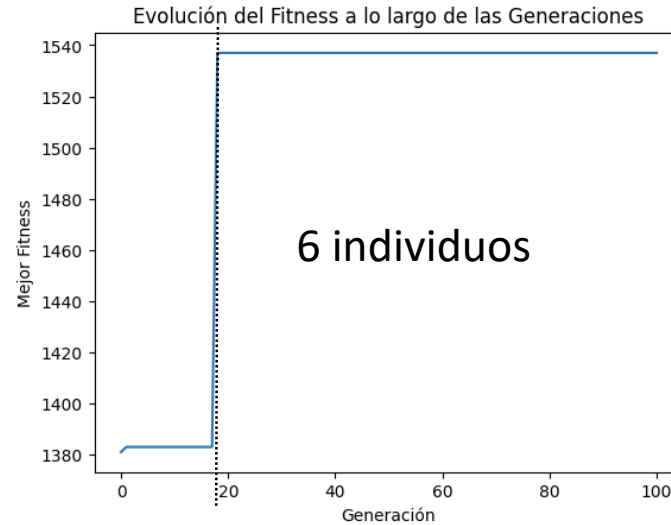
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H002'},



Fitness total final para este individuo: 1537

Generación 100, Mejor fitness = 1537

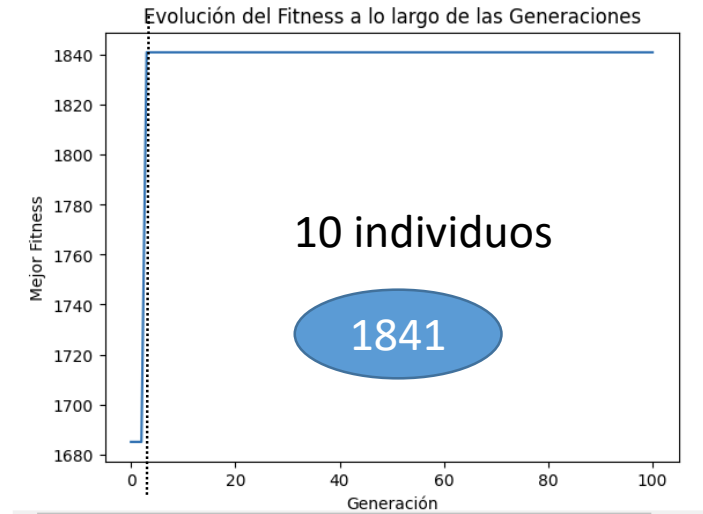
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'}]



Fitness total final para este individuo: 1841

Generación 100, Mejor fitness = 1841

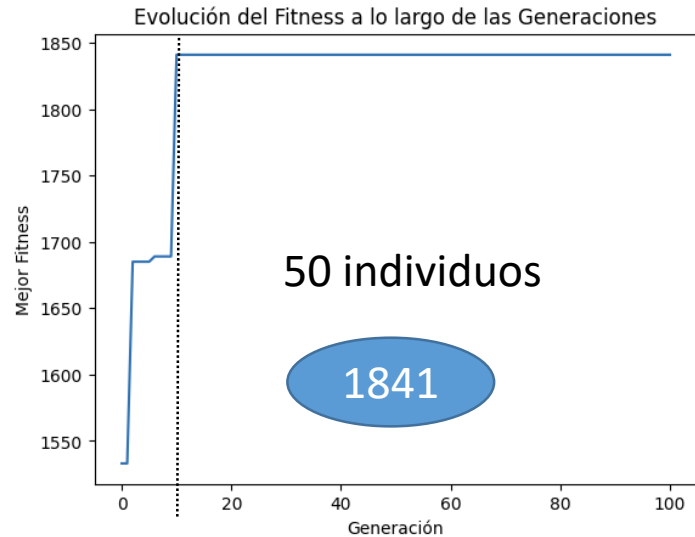
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'}]



Fitness total final para este individuo: 1841

Generación 100, Mejor fitness = 1841

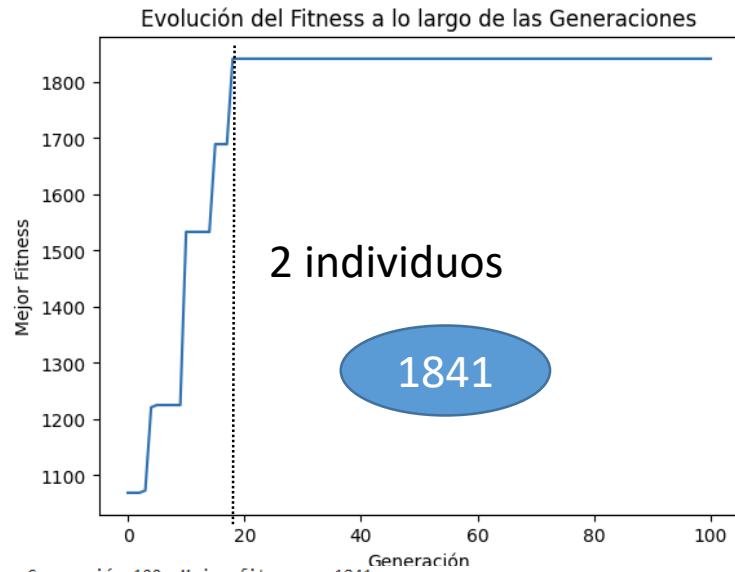
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003'}]



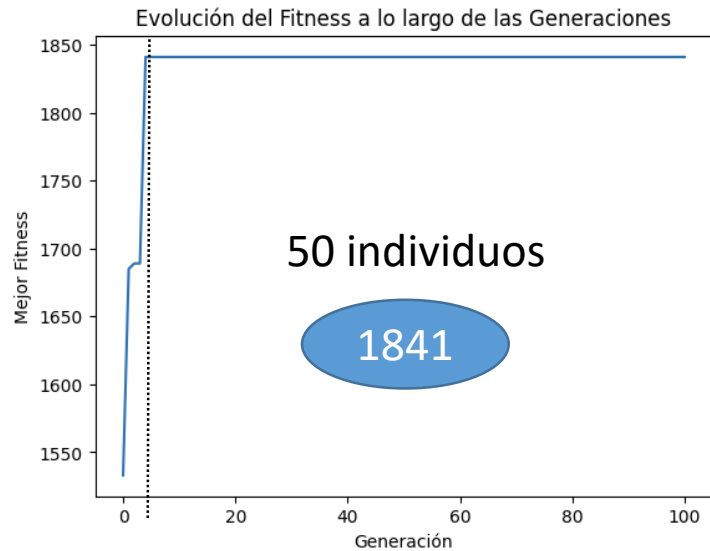
```
4 # Hiperparámetros del algoritmo genético
5 tamaño_poblacion = 2 # número de individuos
6 GENERATIONS = 100 # número de generaciones
7 PMUT = 0.0 # tasa de mutación
8
9 # Inicializa una población inicial de forma aleatoria
10 poblacion_inicial = init_population(tamaño_poblacion, cursos, horarios, aulas, profesores)
11
12 # Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
13 best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
14                                         crossover="onepoint", mutation="multiflip",
15                                         selection_parents_method='roulette',
16                                         selection_survivors_method='ranking')
17
```

## 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': 'H003'}

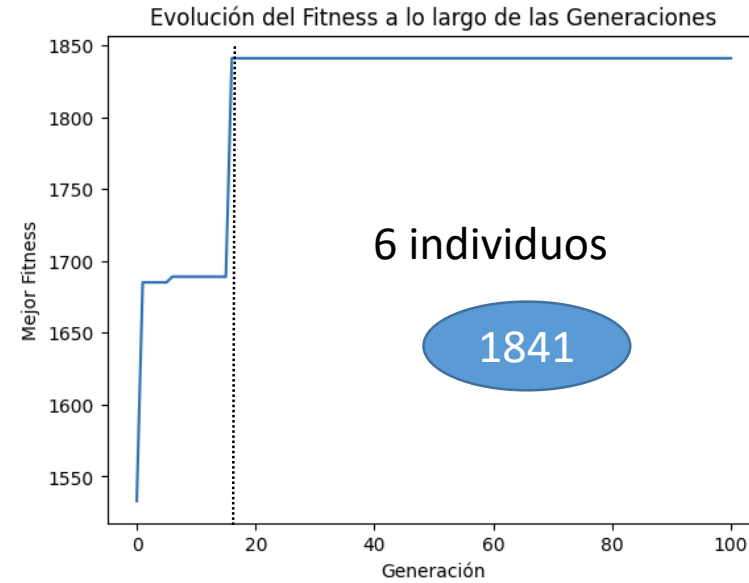


Generación 100, Mejor fitness = 1841  
Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003'}

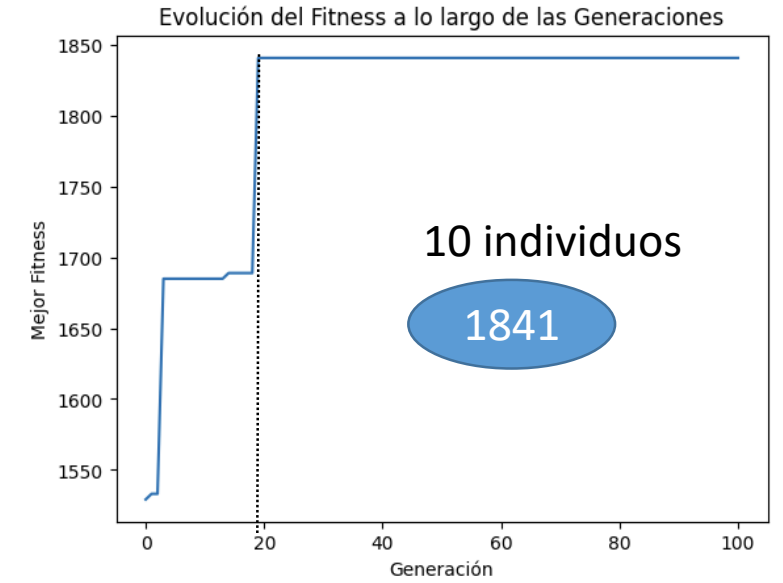


FUENTE: Código Python de Solución de Problema

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



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



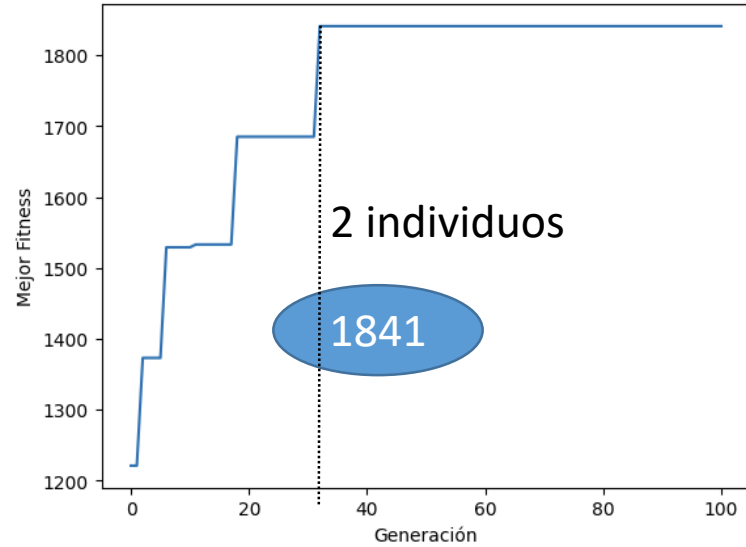
```
9 # Inicializa una población inicial de forma aleatoria
10 poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)
11
12 # Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
13 best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
14 crossover="onepoint", mutation="flip",
15 selection_parents_method='roulette',
16 selection_survivors_method='ranking')
```

## 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

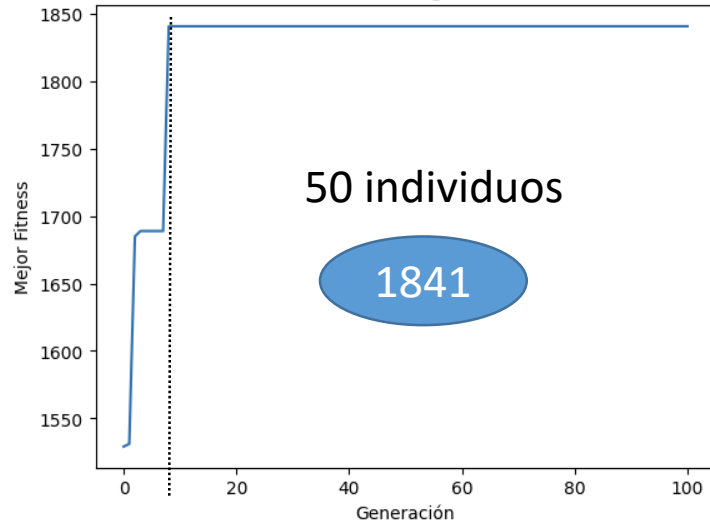
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004

Evolución del Fitness a lo largo de las Generaciones

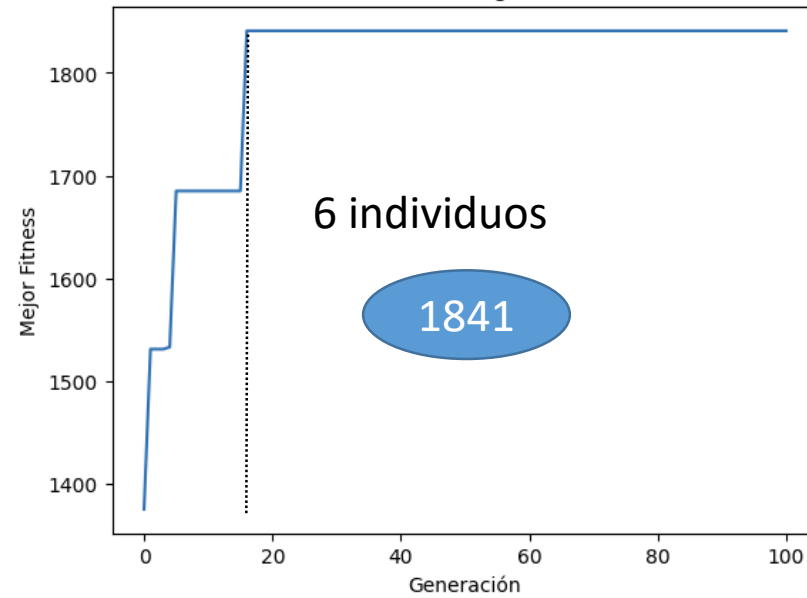


FUENTE: Código Python de Solución de Problema

Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H001'

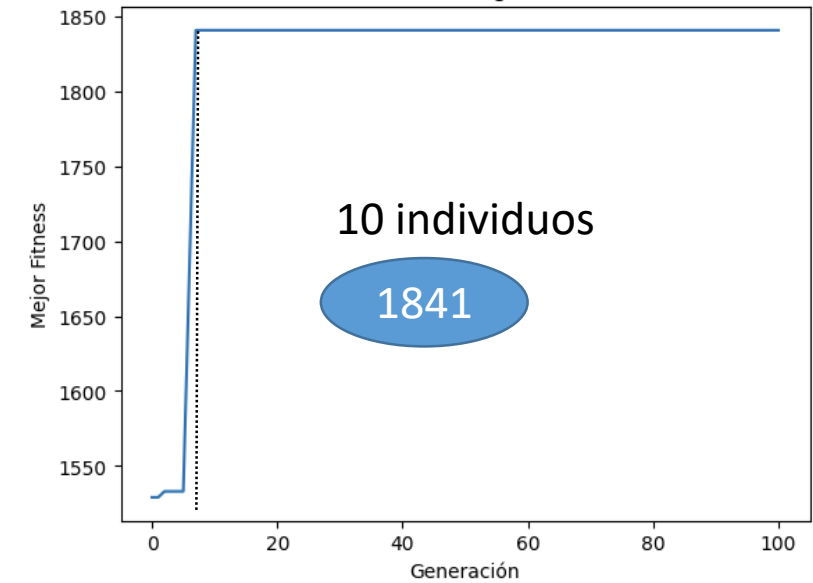
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003

Evolución del Fitness a lo largo de las Generaciones



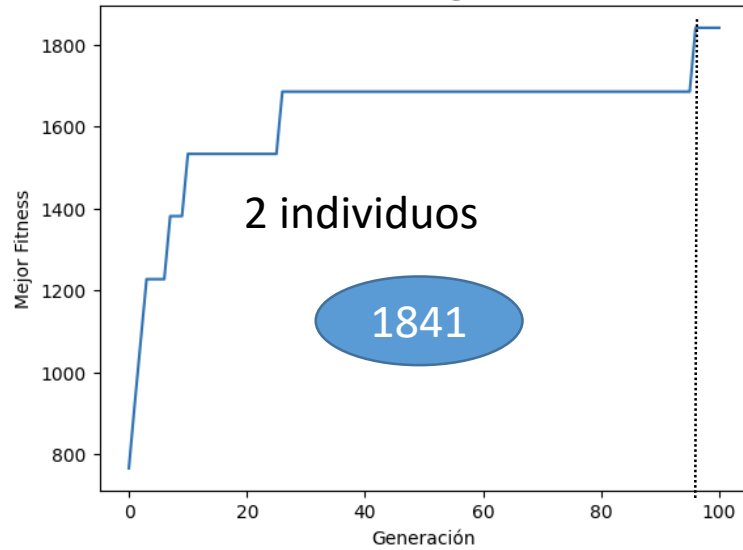
```
9 # Inicializa una población inicial de forma aleatoria
10 poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)
11
12 # Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
13 best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
14                                           crossover="onpoint", mutation="multiflip",
15                                           selection_parents_method='roulette',
16                                           selection_survivors_method='ranking')
```

## 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'}]

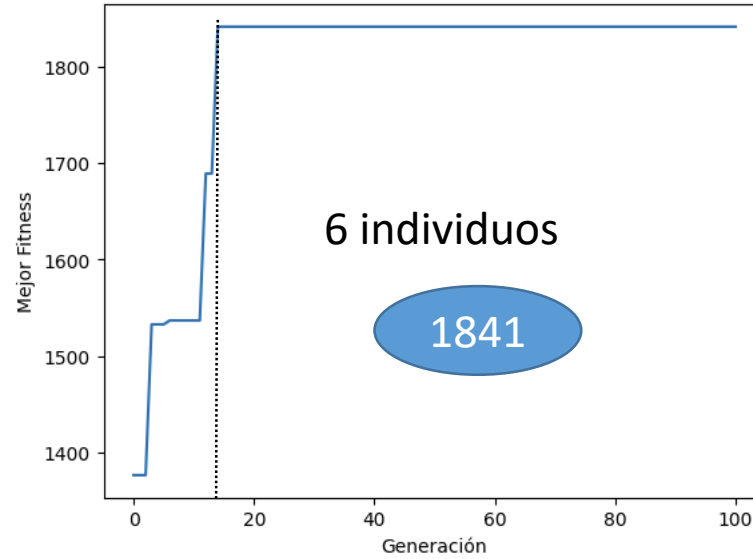
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H004'}]

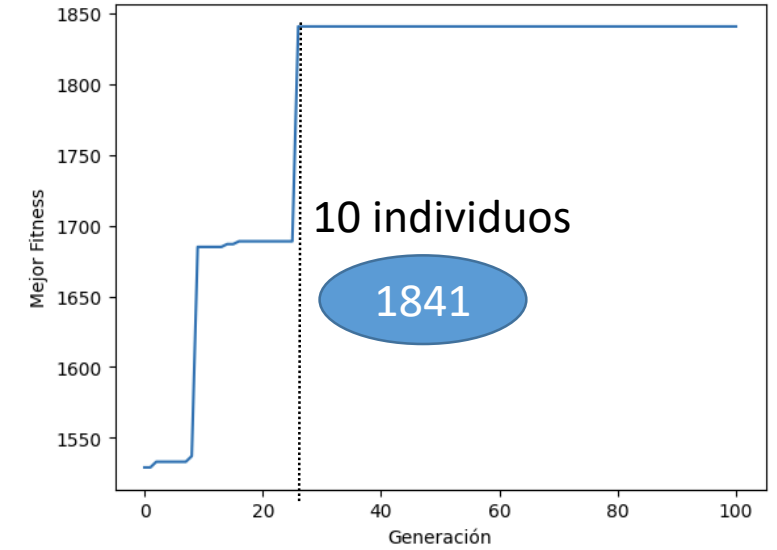
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003'}]

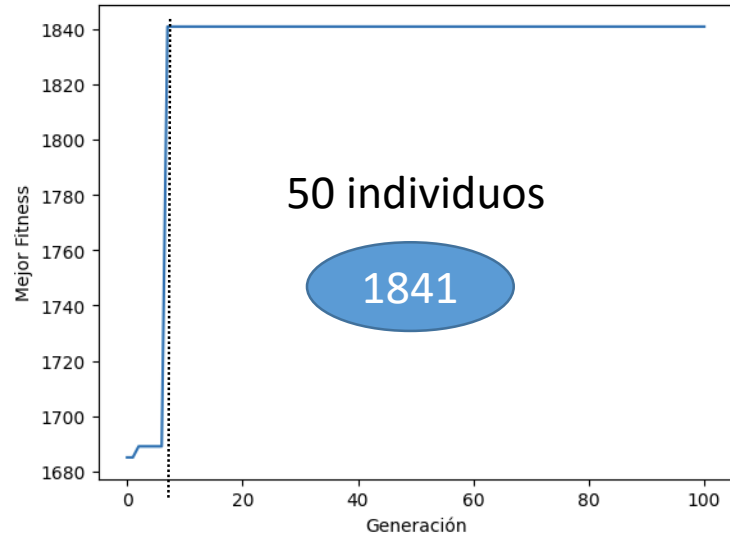
Evolución del Fitness a lo largo de las Generaciones



Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H003'}]

Evolución del Fitness a lo largo de las Generaciones



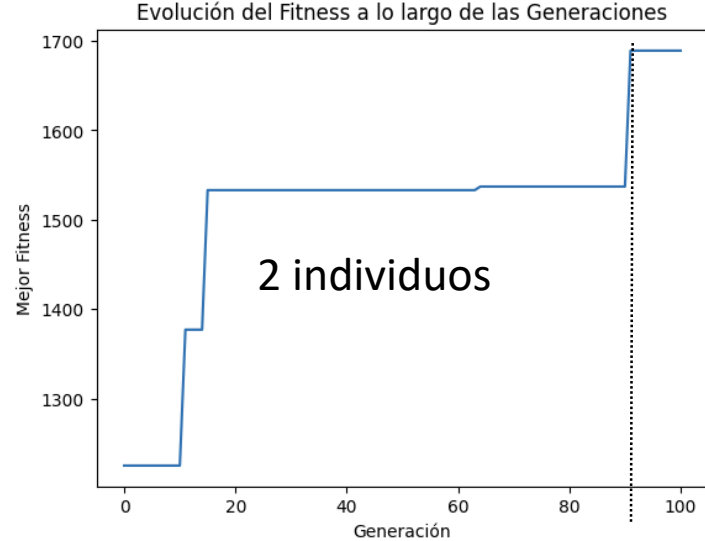
```
9 # Inicializa una población inicial de forma aleatoria
10 poblacion_inicial = init_population(tamano_poblacion, cursos, horarios, aulas, profesores)
11
12 # Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
13 best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
14                                         crossover="uniform", mutation="flip",
15                                         selection_parents_method='roulette',
16                                         selection_survivors_method='ranking')
17
```



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

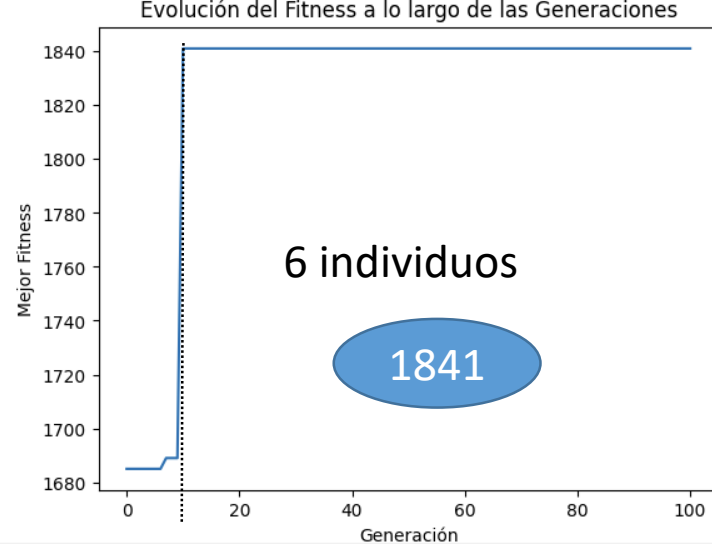
Generación 100, Mejor fitness = 1689

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H00':



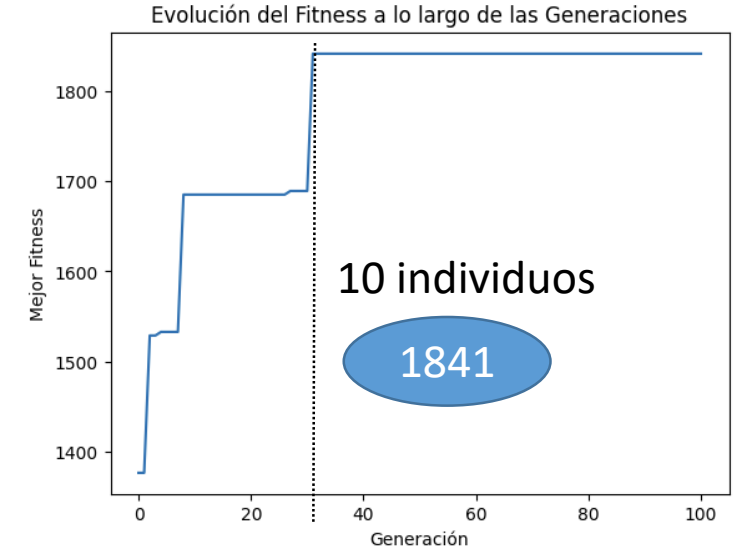
Generación 100, Mejor fitness = 1841

Mejor individuo en la última generación = [{'curso': 'C001', 'horario': 'H002':



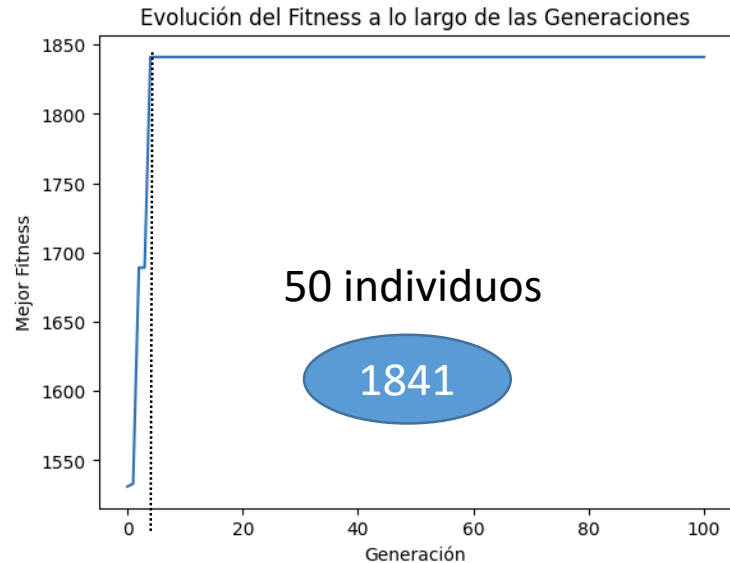
Generación 100, Mejor fitness = 1841

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':



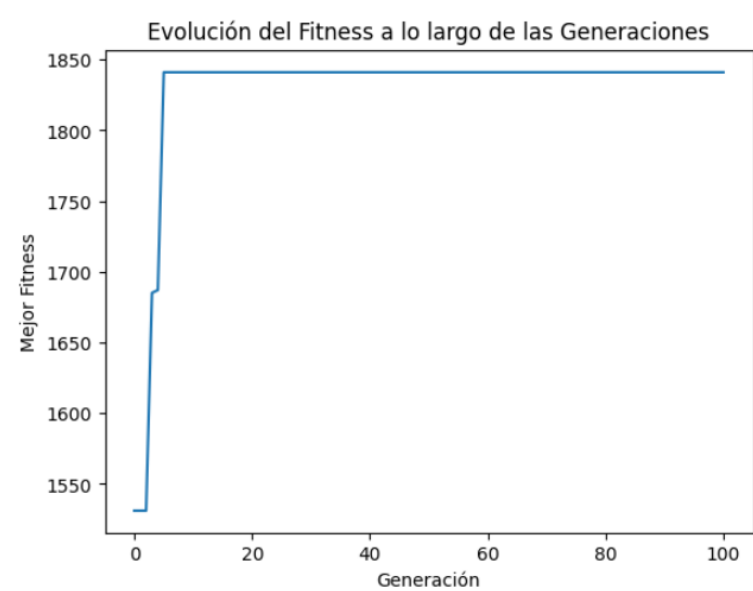
```
# Hiperparámetros del algoritmo genético
tamano_poblacion = 6 # número de individuos
GENERATIONS = 100 # número de generaciones
PMUT = 0.5 # tasa de mutación

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

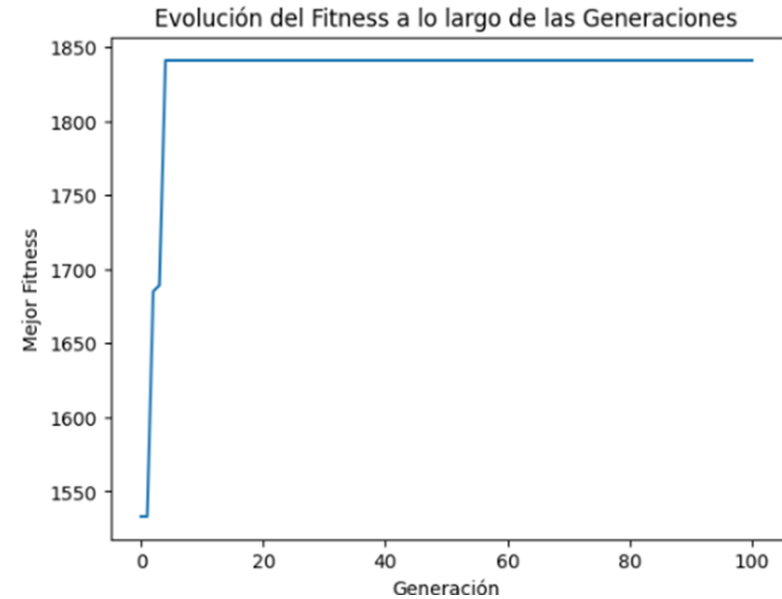
# Evoluciona la población con el algoritmo genético (cruzamiento 'uniforme', mutación 'flip')
best_ind, bestfitness = genetic_algorithm(poblacion_inicial, GENERATIONS, PMUT,
                                         crossover="uniform", mutation="multiflip",
                                         selection_parents_method='roulette',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

MULTIFLIP , 50 INDIVIDUOS , 100 GENERACIONES

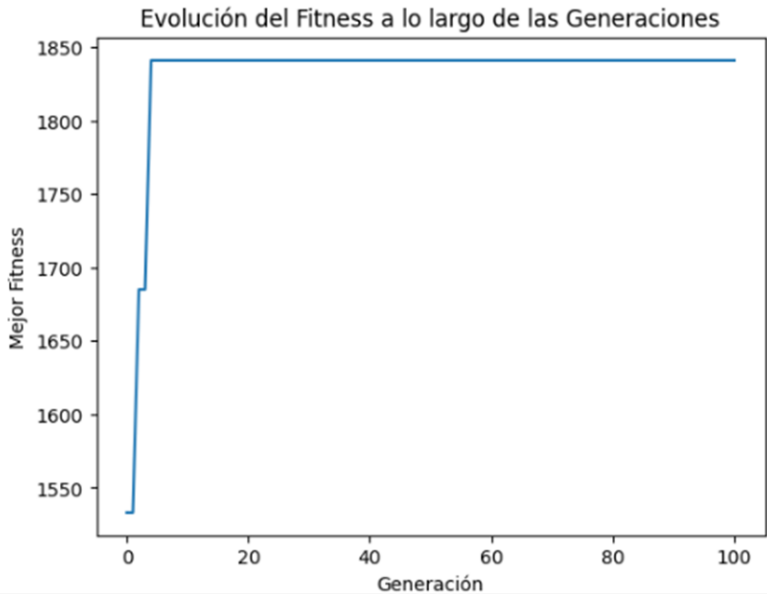
Pmut: 5%



Pmut: 10%

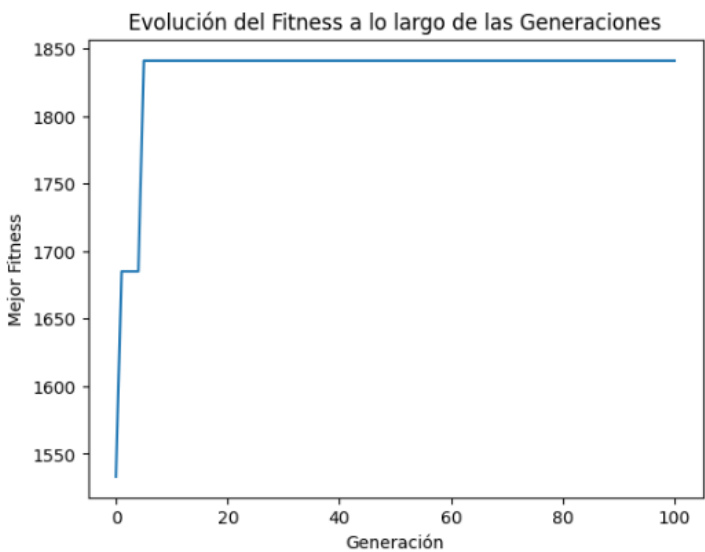


Pmut: 100%

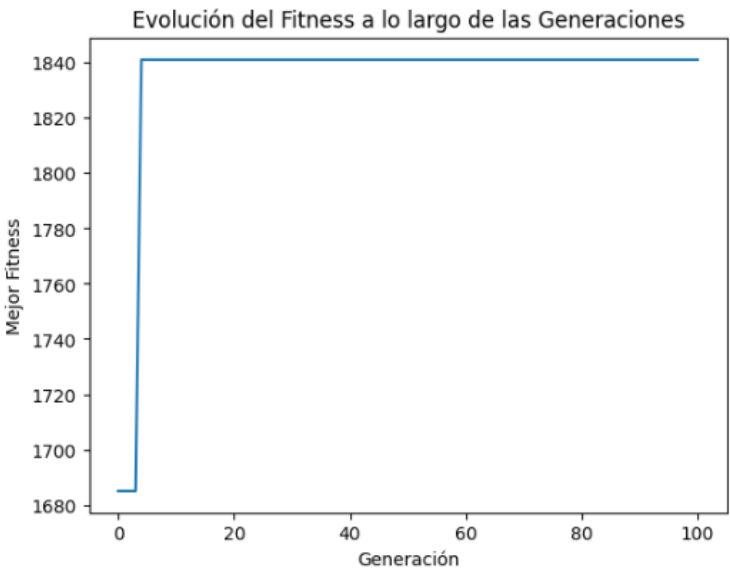


FLIP , 50 INDIVIDUOS , 100 GENERACIONES

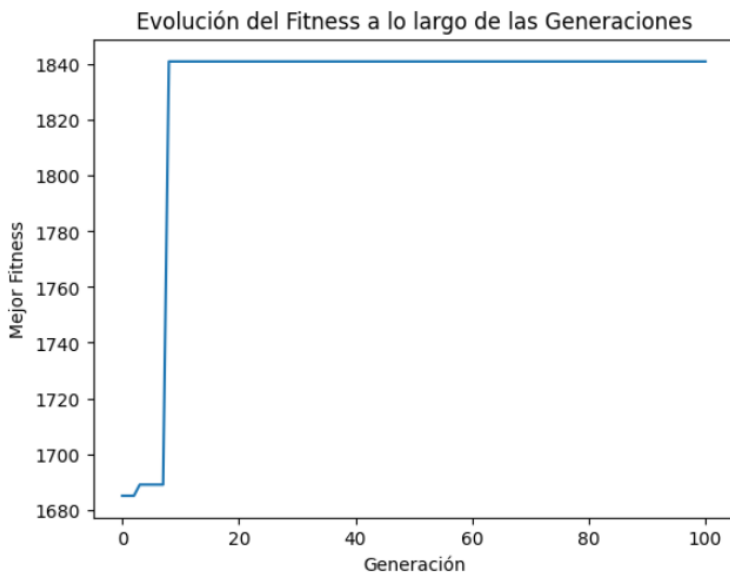
Pmut: 5%



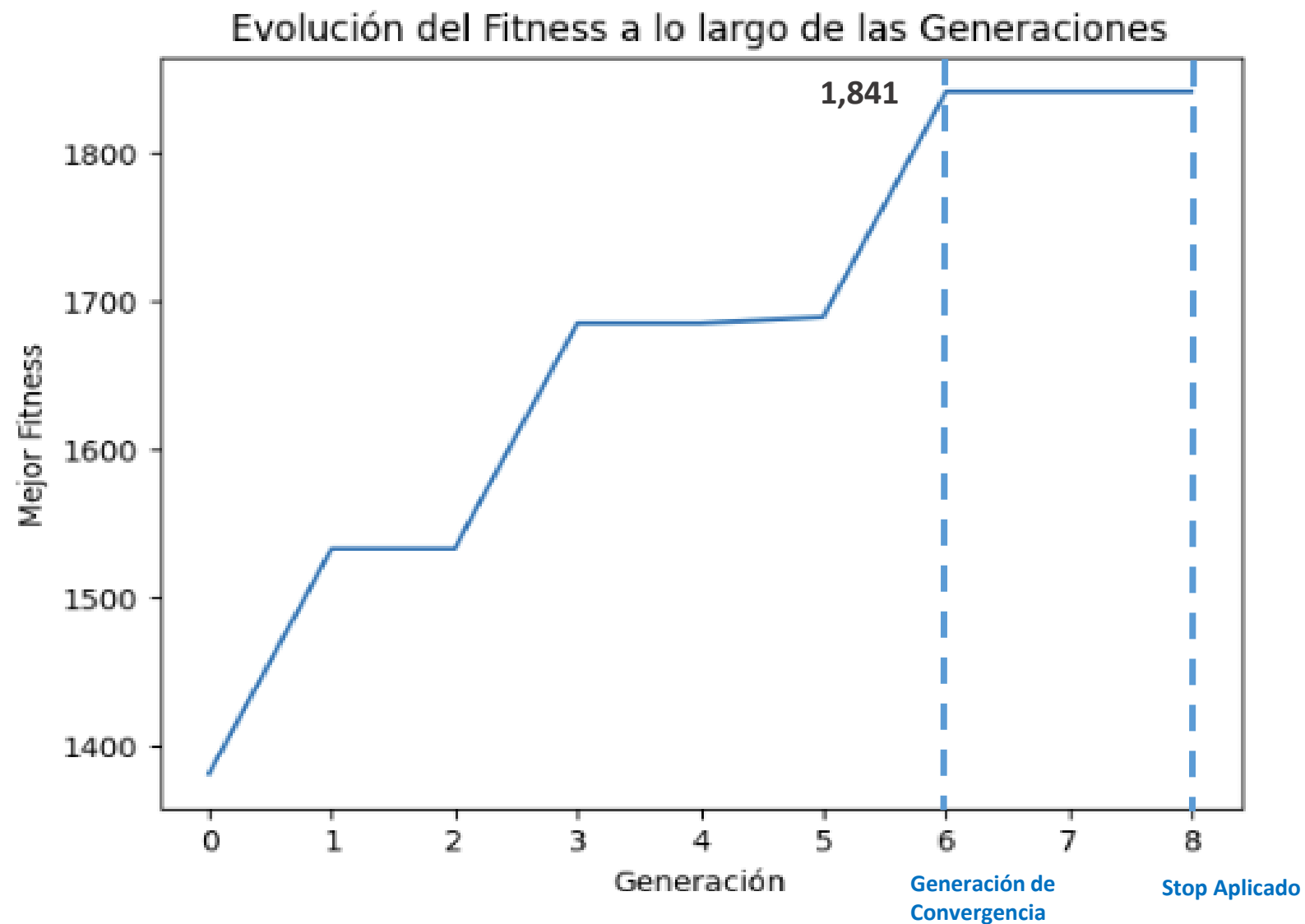
Pmut: 10%



Pmut: 100%



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

*Gracias por su atención !*