Tarea 3.

Cómputo Evolutivo.

Luis Andrés Eguiarte Morett.

20 de abril de 2021

1. Desarrollo del programa.

1.1. Problemas.

Se llevo a cabo un desarrollo en el lenguaje de programación python, cuidando algunos aspectos de la modularidad y escalabilidad del programa, en particular se intentó separar las responsabilidades de los problemas a resolver y el algoritmo genético introduciendo una clase general para los problemas *Problem*, la cual se define a partir de una interfaz *IProblem* con cuatro métodos principales:

1. evaluate: Evalua las soluciones potenciales del problema.
2. stop_criteria: Regresa si la población ha llegado al criterio de paro.
3. populate: Crea una poblacion inicial de posibles soluciones.
4. decode: Pasa a la población del genotipo al fenotipo

La idea principal es desacoplar completamente al algoritmo genético de los problemas que resolverá, dandole toda la responsabilidad a los problemas de inicializar las poblaciones, decodificar y evaluar, de tal manera que el algoritmo genético únicamente se preocupe por aplicar los operadores de selección, cruza y mutación de una manera independiente de la codificación y el problema.
Es así que la estructura principal del problema queda de la siguiente manera:

```
class ProblemInterface(metaclass=abc.ABCMeta):
    @classmethod
    def __subclasshook__(cls, subclass):
        return (hasattr(subclass, 'evaluate') and
                callable(subclass.evaluate) and
                hasattr(subclass, 'stop_criteria') and
                callable(subclass.stop_criteria) and
                hasattr(subclass, 'populate') and
                callable(subclass.populate) and
                hasattr(subclass, 'decode') and
```

```
                callable(subclass.decode))


@ProblemInterface.register
class IProblem:
    """Evalua las soluciones potenciales del problema"""
    def evaluate(self, X):
        pass

    """Regresa si la población ha llegado al criterio de paro"""
    def stop_criteria(self, X_eval):
        pass

    """Crea una poblacion inicial de posibles soluciones"""
    def populate(self, n_individuals):
        pass

    """Pasa a la población del genotipo al fenotipo"""
    def decode(self, X_encoded):
        pass
```

1.2. Algoritmo genético.

En cuanto al algoritmo genético, se programó en una clase base *GeneticAlgorithm*, que por el momento implementa los diferentes criterios de selección con una serie de bloques if-else y un par de propiedades que indican el porcentaje de elitismo *elitism* y el tipo de selección *selection*, en el futuro se implementará con una herarquía de clases adecuada para habilitar una mayor modularidad y escalabilidad.
Respecto a la propiedad *elitism*, indica que porcentaje de la población con mayor aptitud se mantendrá intacto para la siguiente generación, estos individuos serán considerados para los padres para la siguiente generación. Una vez que se coloca a los individuos élite en la población de la generación siguiente, se efectúa una selección a través del criterio que indique la propiedad *selection*, que por los requerimientos del programa será una de entre proporcional (*proportional*), torneo (*tournament*) y sobrante estocástico universal (*sus*).

1.2.1. Definición de la clase.

La definición de la clase *GeneticAlgorithm*, queda de la siguiente forma:

```
class GeneticAlgorithm:
    def __init__(self, n_individuals = 10, pc = 0.6, pm = 0.1, max_iter = 5000,
                 elitism = None, selection = "proportional"):
```

```
        self.max_iter = max_iter
        self.n_individuals = n_individuals
        self.pc = pc
        self.pm = pm
        self.elitism = elitism
        self.selection = selection



    """
    En este método se implementan el algoritmo genético principal,
    con su ciclo de ejecución, mandando a llamar a los operadores genéticos
    """
    def evolve(self, problem):


    """
    En este método se implementa el criterio de mutación para cada uno de los genes
    del cromosoma de cada uno de los individuos de la población
    """
    def mutate(self, X):

    """
    En este método se lleva a cabo la cruza de un punto tomando un par de individuos
    de los que fueron seleccionados a la vez y tomando en cuenta la probabilidad
    de mutación
    """
    def crossover(self, X):

    """
    En este método se lleva a cabo la selección en cada una de sus variantes
    - Proporcional c/s elitismo
    - Torneo c/s elitismo
    - SUS c/s elitismo
    """
    def select(self, problem, X):
```

Ahora se procede a comentar cada uno de los métodos definidos anteriormente.

1.2.2. Método principal.

Éste método se reduce a que se le pase un objeto de la clase *Problem*, y realizar iterativamente las operaciones de selección, cruza y mutación. El criterio de paro es ya sea llegando al número máximo de iteraciones o que se llegue a un criterio de paro definido por el problema que se le esté pasando, este criterio se define en cada una de las subclases que

implementan los problemas particulares. Lo que se hizo para cada una de las funciones fue tomar el máximo valor dentro del rango definido de la función y restarle el valor de la función evaluada, esto con el objetivo de poder efectuar las selecciones proporcionales y SUS.

```
    def evolve(self, problem):
        its = 1
        self.pop = problem.populate(self.n_individuals)
        while its <= self.max_iter:
            self.pop, solution = self.select(problem, self.pop)
            if len(solution) > 0:
                print(its)
                return problem.decode(self.pop[solution, :])
            self.pop = self.crossover(self.pop)
            self.pop = self.mutate(self.pop)
            its+=1
        eval_X = problem.evaluate(self.pop)

        eval_X = np.sort(eval_X, order = ['fitness'])['index']
        return problem.decode(self.pop[eval_X[0:2], :])
```

En este método se implementan cada uno de los criterios de selección, tanto con elitismo como sin elitismo.
La selección proporcional se efectúa obteniendo las probabilidades de selección para cada individuo como la aptitud del individuo entre la suma de las aptitudes de toda la población, posteriormente se efectúa la suma acumulada de las probabilidades para que al obtener un número aleatorio entre 0 y 1 para cada uno de los elementos de la población, se vea reflejado el efecto de girar una ruleta con probabilidades escaladas proporcionalmente a la aptitud de los individuos, dado que el arreglo con sumas acumuladas está ordenado, para cada número aleatorio generado, éste se puede buscar mediante una búqueda binaria y así obtener el índice del elemento elegido, reduciendo la complejidad de este algoritmo de la selección por ruleta de $O(n^2)$ a $O(nlog(n))$.

La selección por torneo se efectúa en un esquema de torneo entre pares, donde ambas mitades de los padres de la siguiente generación se obtienen generando una permutación de n elementos, donde n es el tamaño de la población, tomando dos elementos cada vez y seleccionando como padre el que tenga mayor aptitud.

La selección por SUS se lleva a cabo obteniendo los valores esperados de los padres de la siguiente generación y por cada uno de los valores asignar un puntero a una sección de la ruleta con cierta aleatoriedad asociada a cada valor esperado de fitness, generando la suma acumulada de los fitness en un arreglo, creamos nuestra ruleta, lo que se hizo al generar el arreglo de punteros anterior, fue para fines prácticos como girar la ruleta el número de veces necesario, por separado. Después, por cada valor del puntero se hace una búsqueda binaria sobre el arreglo de fitness acumulado, para obtener el índice del padre seleccionado.

Todos los esquemas de selección anteriores se mantienen para el elitismo, lo único que cambiará es que se obtendrán $n - \lfloor e * n \rfloor$ posibles padres a través del esquema de selección elegido para la siguiente generación, a los cuales se les podrá aplicar los operadores de cruza y mutación para producir $n - \lfloor e * n \rfloor$ hijos que serán miembros de la población de la siguiente generación. A los $\lfloor e * n \rfloor$ mejores individuos no se les aplicarán estos operadores y se pasarán tal cual a la siguiente generación. e es el parámetro correspondiente al porcentaje de elitismo.
Hay que hacer notar que los padres de la siguiente generación se pueden seguir eligiendo de entre los individuos de la élite.

1.2.3. Selección.

```
    def select(self, problem, X):
        eval_X = problem.evaluate(X)
        solution = problem.stop_criteria(eval_X['fitness'])

        if len(solution) > 0:
            return X, solution

        if not self.elitism:
            if self.selection == "proportional":
                eval_X = np.sort(eval_X, order = ['fitness'])
                prob_sel = eval_X['fitness']/eval_X['fitness'].sum()
                c_prob_sel = np.cumsum(prob_sel)
                probs = np.random.rand(1,X.shape[0])
                new_X = np.zeros(X.shape, dtype = np.int64)
                for j, prob in enumerate(probs[0, :]):
                    i = np.searchsorted(c_prob_sel, prob)
                    new_X[j, :] = X[eval_X['index'][i], :]
                return new_X, solution
            elif self.selection == "tournament":
                new_X = np.zeros(X.shape, dtype = np.int64)
                for i in range(2):
                    perm = np.random.permutation(X.shape[0])
                    for j in range(0, X.shape[0]//2):
                        index = [perm[2 * j], perm[2 * j + 1]]
                        max_ind =
                        np.argmax([eval_X['fitness'][index[0]],
                                   eval_X['fitness'][index[1]]])
                        new_X[i * X.shape[0]//2 + j, :] = X[index[max_ind], :]
                return new_X, solution
            elif self.selection == "sus":
                prob = eval_X['fitness'].sum()/X.shape[0]
                pointers = [random.random()*prob + i*prob for i in range(X.shape[0])]
```

```
            new_X = np.zeros(X.shape, dtype = np.int64)
            cum_fitness = np.cumsum(eval_X['fitness'])
            for j, p in enumerate(pointers):
                i = np.searchsorted(cum_fitness, p)
                new_X[j, :] = X[i, :]
            return new_X, solution


    else:
        elitism_num = math.floor(self.elitism * X.shape[0])
        if self.selection == "proportional":
            eval_X = np.sort(eval_X, order = ['fitness'])
            prob_sel = eval_X['fitness']/eval_X['fitness'].sum()
            c_prob_sel = np.cumsum(prob_sel)
            probs = np.random.rand(1,X.shape[0] - elitism_num)
            new_X = np.zeros(X.shape, dtype = np.int64)
            new_X[0 : elitism_num, :] =
            X[eval_X['index'][eval_X.size - elitism_num: eval_X.size], :]
            for j, prob in enumerate(probs[0, :]):
                i = np.searchsorted(c_prob_sel, prob)
                new_X[j + elitism_num, :] = X[eval_X['index'][i], :]
            return new_X, solution
        elif self.selection == "tournament":
            new_X = np.zeros(X.shape, dtype = np.int64)
            new_X[0 : elitism_num, :] =
            X[eval_X['index'][eval_X.size - elitism_num: eval_X.size], :]
            #print(eval_X)
            for i in range(2):
                perm = np.random.permutation(X.shape[0])
                #print(perm)
                for j in range(0, (X.shape[0] - elitism_num)//2):
                    index = [perm[2 * j], perm[2 * j + 1]]
                    max_ind = np.argmax([eval_X['fitness'][index[0]],
                    eval_X['fitness'][index[1]]])
                    new_X[i * (X.shape[0] - elitism_num)//2 + j + elitism_num, :] =
                    X[index[max_ind], :]
            return new_X, solution
        elif self.selection == "sus":
            prob = eval_X['fitness'].sum()/(X.shape[0] - elitism_num)
            pointers =
            [random.random()*prob + i*prob for i in range(X.shape[0] - elitism_num)]
            new_X = np.zeros(X.shape, dtype = np.int64)
            new_X[0 : elitism_num, :] =
            X[eval_X['index'][eval_X.size - elitism_num: eval_X.size], :]
            cum_fitness = np.cumsum(eval_X['fitness'])
```

```
            for j, p in enumerate(pointers):
                i = np.searchsorted(cum_fitness, p)
                new_X[j + elitism_num, :] = X[i, :]
            return new_X, solution
```

1.2.4. Cruza.

Respecto a la cruza, se sigue un esquema de cruza de un punto con probabilidad de cruza p_c entre cada par de padres. Para cada par de padres se obtiene un número aleatorio r_i, para cada par i-ésimo que cumpla con $r_i \leq p_c$, se efectuará una cruza que producirá dos hijos, y se obtendrá un punto de cruza aleatorio para llevarla a cabo.

```
    def crossover(self, X):
        if not self.elitism:
            n_cross = self.n_individuals // 2
            elitism_num = 0
        else:
            elitism_num = math.floor(self.elitism * X.shape[0])
            n_cross = (self.n_individuals - elitism_num) // 2
        prob_cross = np.random.rand(1 , n_cross)[0,:]
        for i, p in enumerate(prob_cross):
            if p <= self.pc:
                cross_point = random.randint(1, X.shape[1])
                # print(i)
                # print(cross_point)
                son1 = X[2*i + elitism_num,:].copy()
                son2 = X[2*i + 1 + elitism_num, :].copy()
                son1[cross_point : -1] = X[2*i + 1 + elitism_num, cross_point : -1]
                son2[cross_point : -1] = X[2*i + elitism_num, cross_point : -1]
                # print(son1)
                # print(son2)
                X[2*i + elitism_num,:] = son1
                X[2*i + 1 + elitism_num,:] = son2
        return X
```

1.2.5. Mutación.

En cuanto a la mutación, se generara una matriz de números aleatorios de dimensiones $(n, \sum_{i=1}^{d} l_i)$, donde n corresponde al número de padres, d el número de dimensiones del problema y l_i la longitud de la cadena de representación de cada dimensión, cada elemento de esa matriz que cumpla con $M_{ij} \leq p_m$, nos representará una mutación y por lo tanto una inversión de bits, esta inversión se llevará a cabo aplicando la XOR entre los padres y la matriz $L_{ij} = M_{ij} \leq p_m$.

```
def mutate(self, X):
    mutate_m = np.random.rand(X.shape[0], X.shape[1])
    mutate_m = mutate_m <= self.pm
    X_bit = X == 1
    if not self.elitism:
        X = np.logical_xor(X_bit, mutate_m)
    else:
        elitism_num = math.floor(self.elitism * X.shape[0])
        X[X.shape[0] - elitism_num :  X.shape[0], :] =
        np.logical_xor(X_bit, mutate_m)[X.shape[0] - elitism_num :  X.shape[0], :]
    X = X.astype(int)
    return X
```

Recordemos que cuando hablamos de padres, tanto para mutación, como anteriormente, para cruza, nos referimos a los individuos que serán sujetos a la mutación, si hablamos de elitismo, serán todos aquellos que hayan sido seleccionados fuera de los de la élite.

2. Resultados.

Para todos los siguientes experimentos se siguieron los siguientes parámetros y se efectuaron 5 corridas.

2.1. Probabilidades de cruza, tamaños de las poblaciones

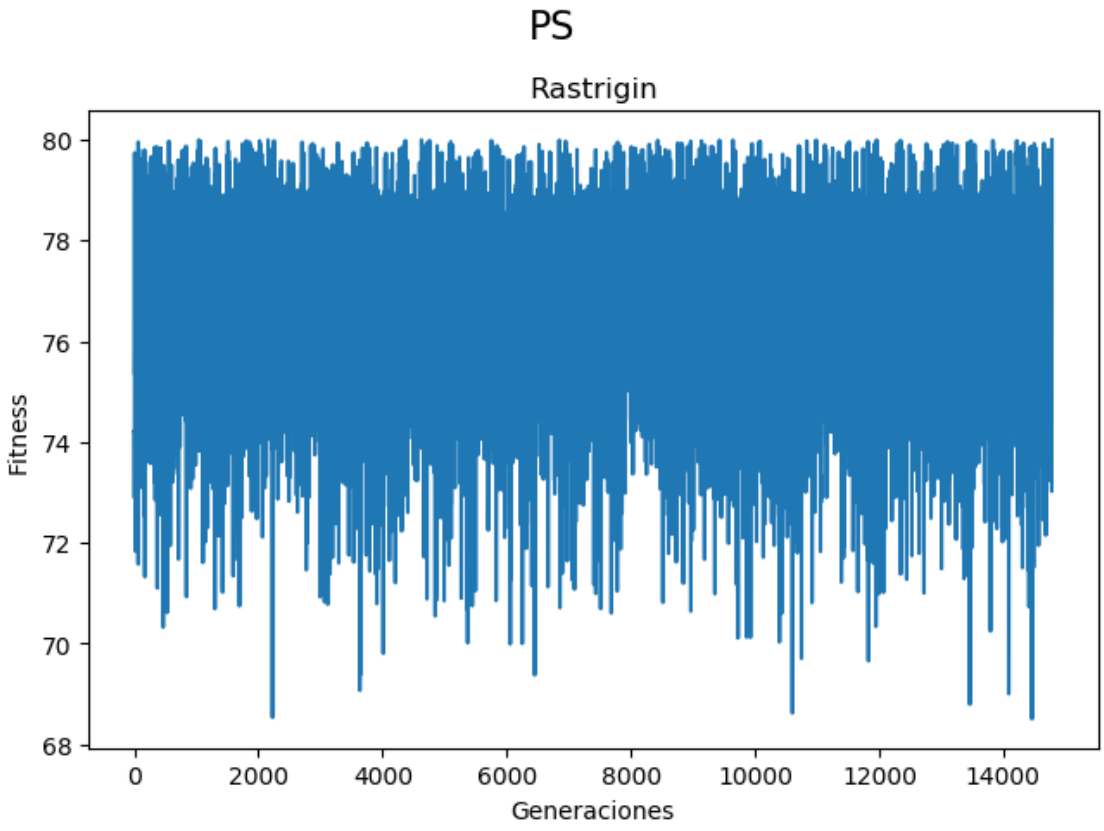
Todas las probabilidades de cruza se mantuvieron iguales para cada función a optimizar.
Rastrigin - 0.9, Población - 100
Beale - 0.95, Población - 100
Himmelblau - 0.95, Población - 100
Eggholder - 0.95, Población - 500

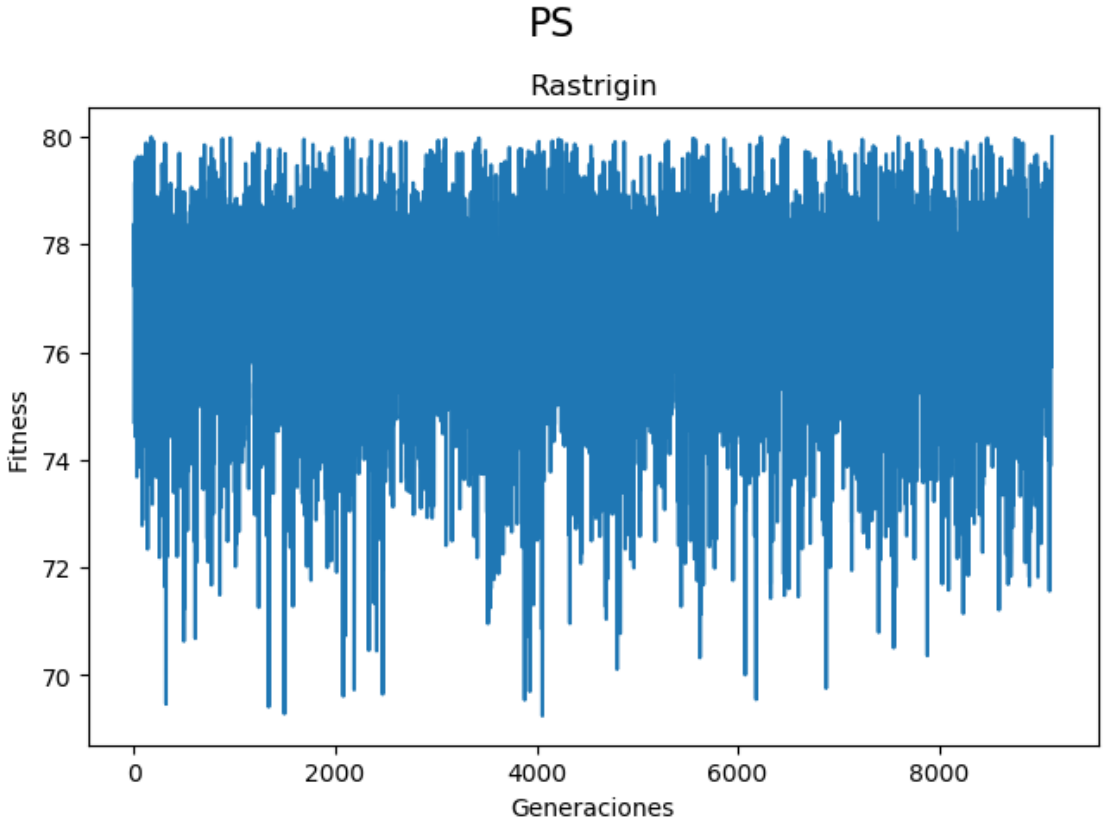
2.2. Selección proporcional, sin elitismo

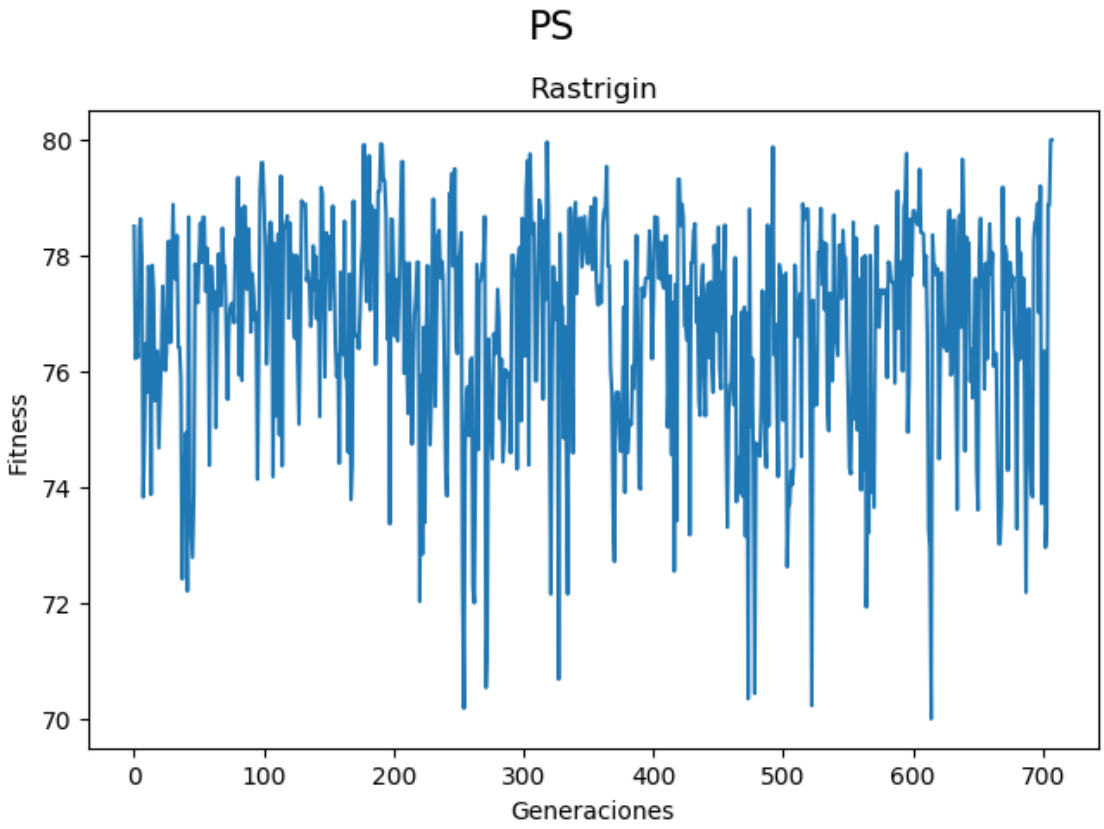
2.2.1. Rastrigin

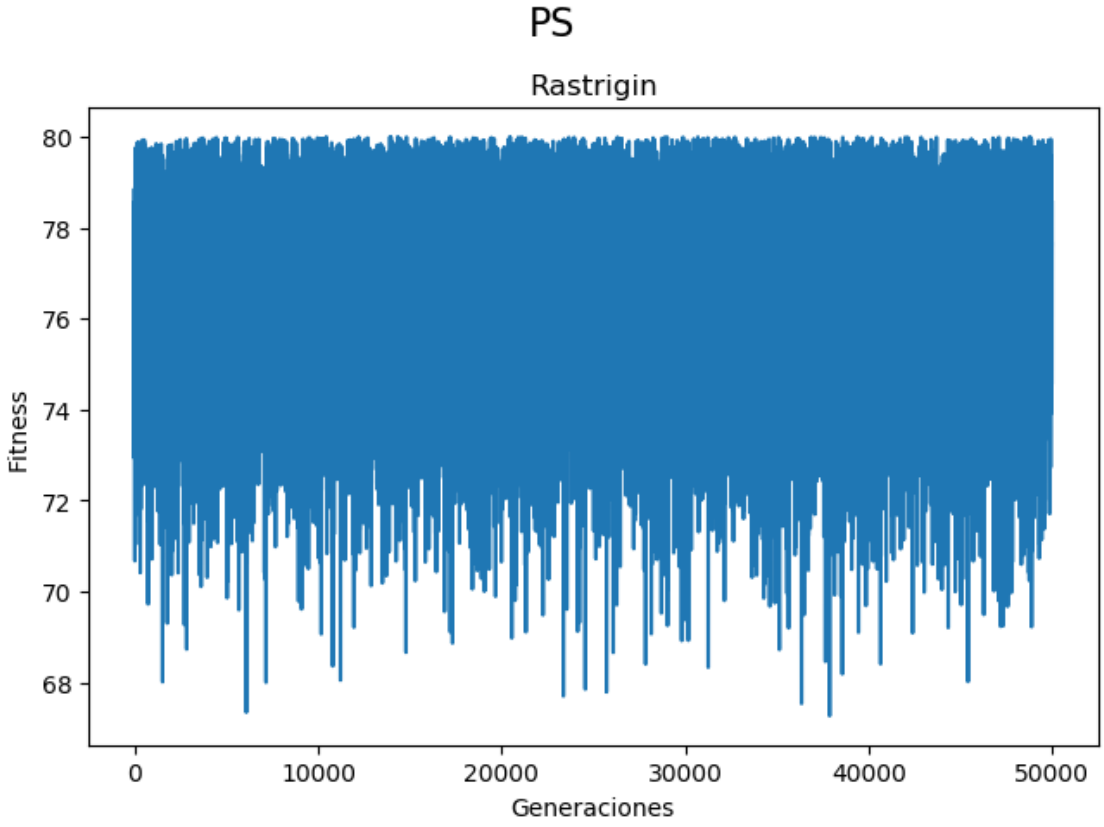
Run: 0
Selection method: PS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[0.00035157 0.0019922]]
after: 14789 generations.
Run: 1
Selection method: PS
Function to optimize: Rastrigin

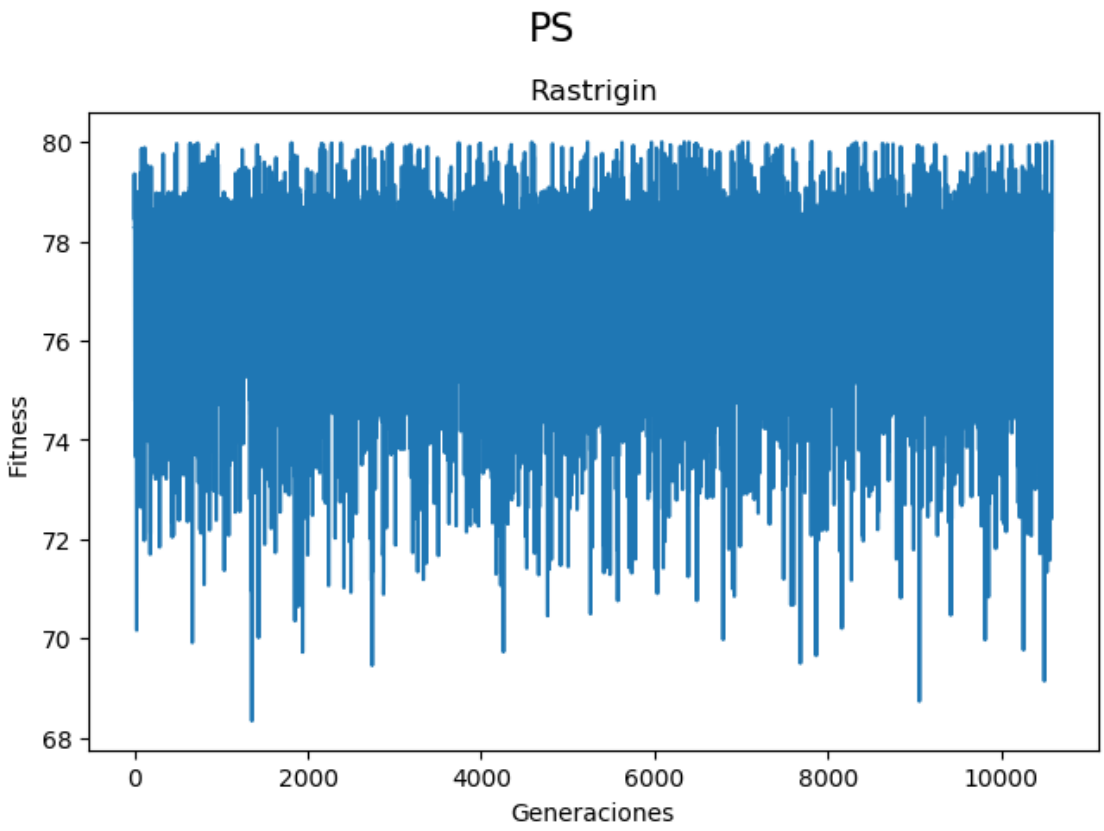
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[-0.0013672 -0.0010547]]
after: 9126 generations.
Run: 2
Selection method: PS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[0.00027344 -0.00113282]]
after: 708 generations.
Run: 3
Selection method: PS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[4.70109055 2.36497898] [-4.56374576 0.70176317]]
after: 50000 generations.
Run: 4
Selection method: PS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[0.00035157 0.00066407]]
after: 10588 generations.


PS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
0
2000
4000
6000
8000
10000
12000
14000


PS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
2000
4000
6000
8000


PS
Rastrigin
Fitness
80
78
76
74
72
70
0
100
200
300
400
500
600
700
Generaciones

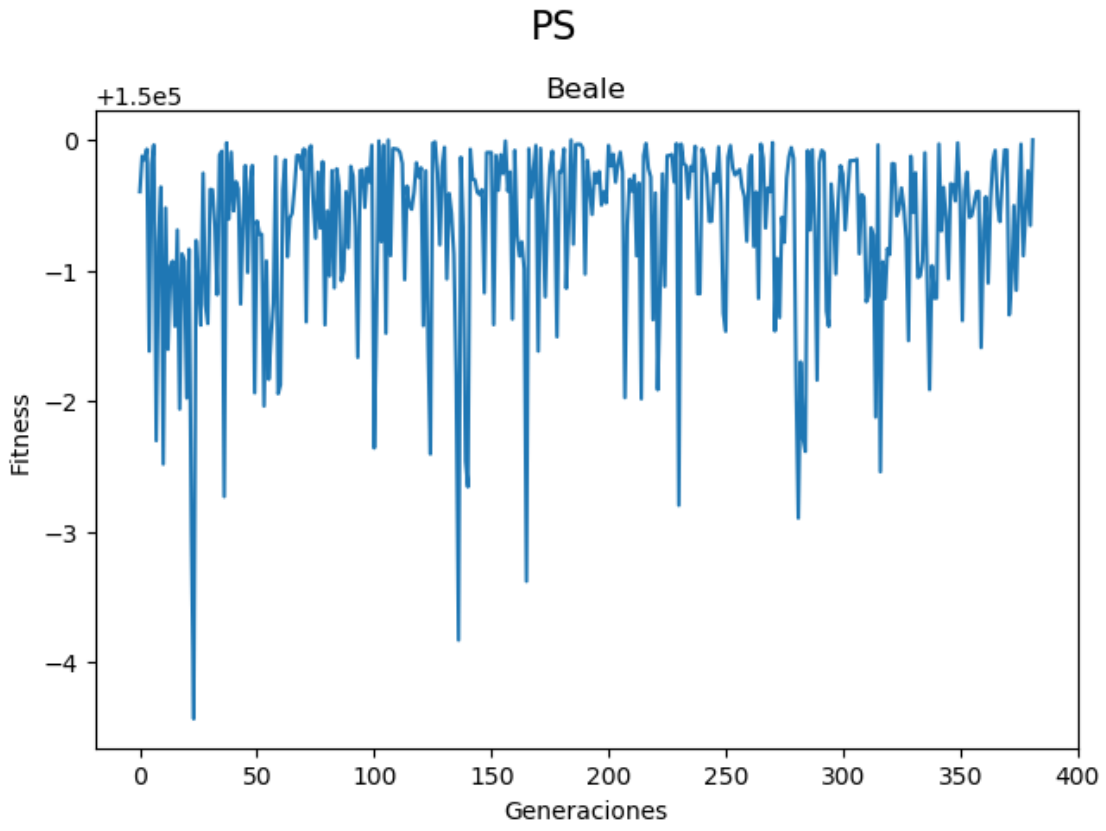

PS
Rastrigin
Fitness
80
78
76
74
72
70
68
0
10000
20000
30000
40000
50000
Generaciones

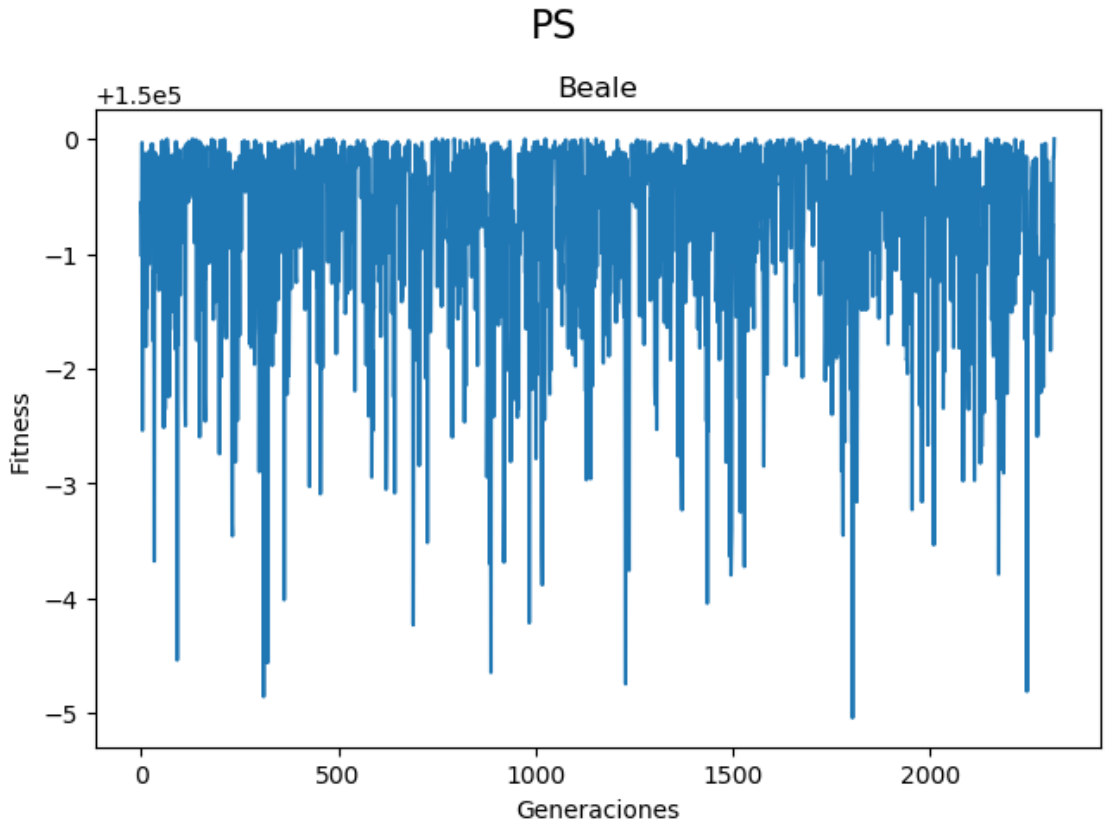


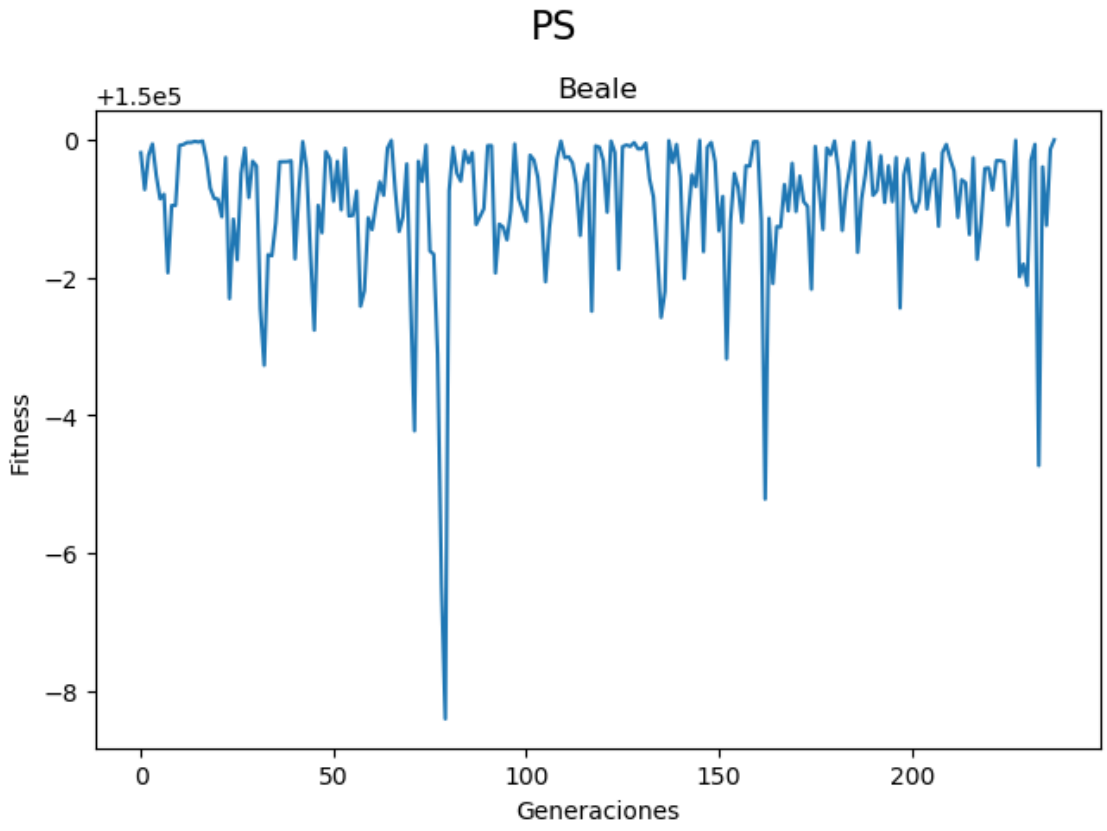
2.2.2. Beale

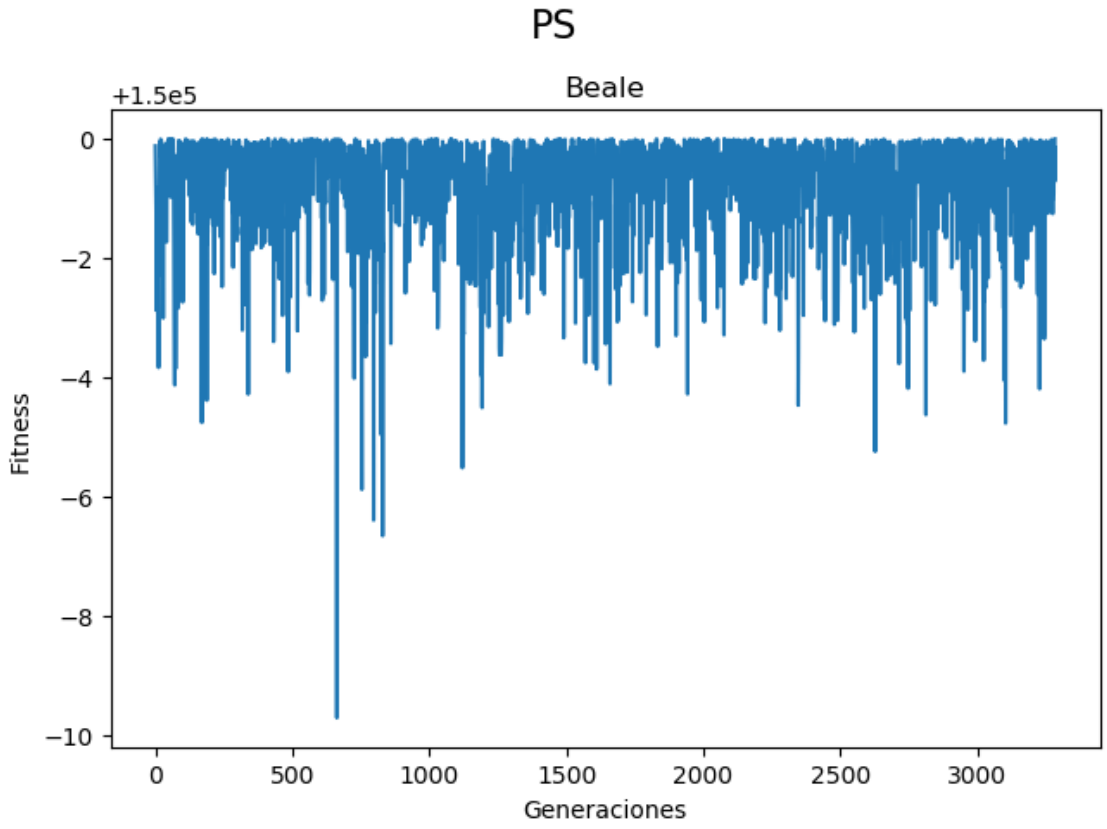
Run: 0
Selection method: PS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[3.00735861 0.50245668]]
after: 382 generations.
Run: 1
Selection method: PS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[2.990467 0.49840544]]
after: 2315 generations.
Run: 2
Selection method: PS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[2.98929969 0.49634549]]
after: 238 generations.
Run: 3
Selection method: PS

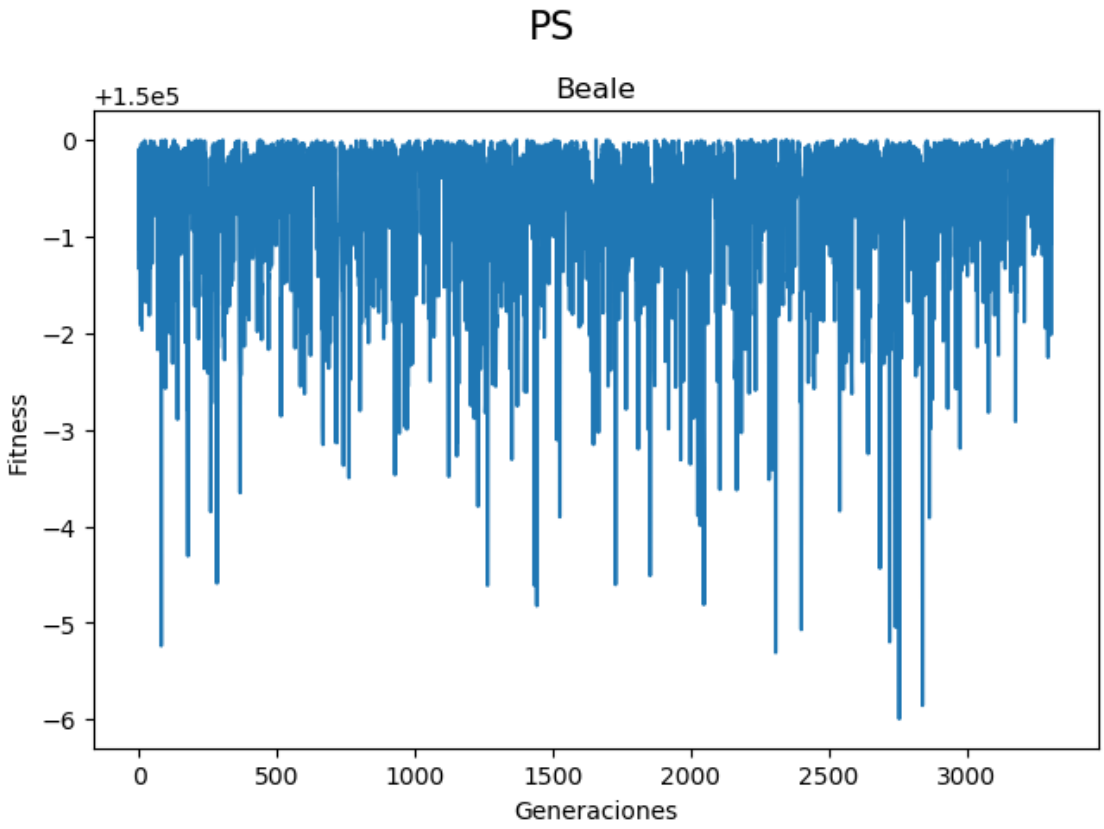
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[2.98284518 0.49655149]]
after: 3285 generations.
Run: 4
Selection method: PS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[2.98119721 0.49483486]]
after: 3310 generations.


PS
Beale
+1.5e5
0
−1
−2
−3
−4
Fitness
0
50
100
150
200
250
300
350
400
Generaciones


PS
Beale
+1.5e5
0
−1
−2
−3
−4
−5
Fitness
0
500
1000
1500
2000
Generaciones


PS
Beale
+1.5e5
0
−2
−4
−6
−8
Fitness
0
50
100
150
200
Generaciones

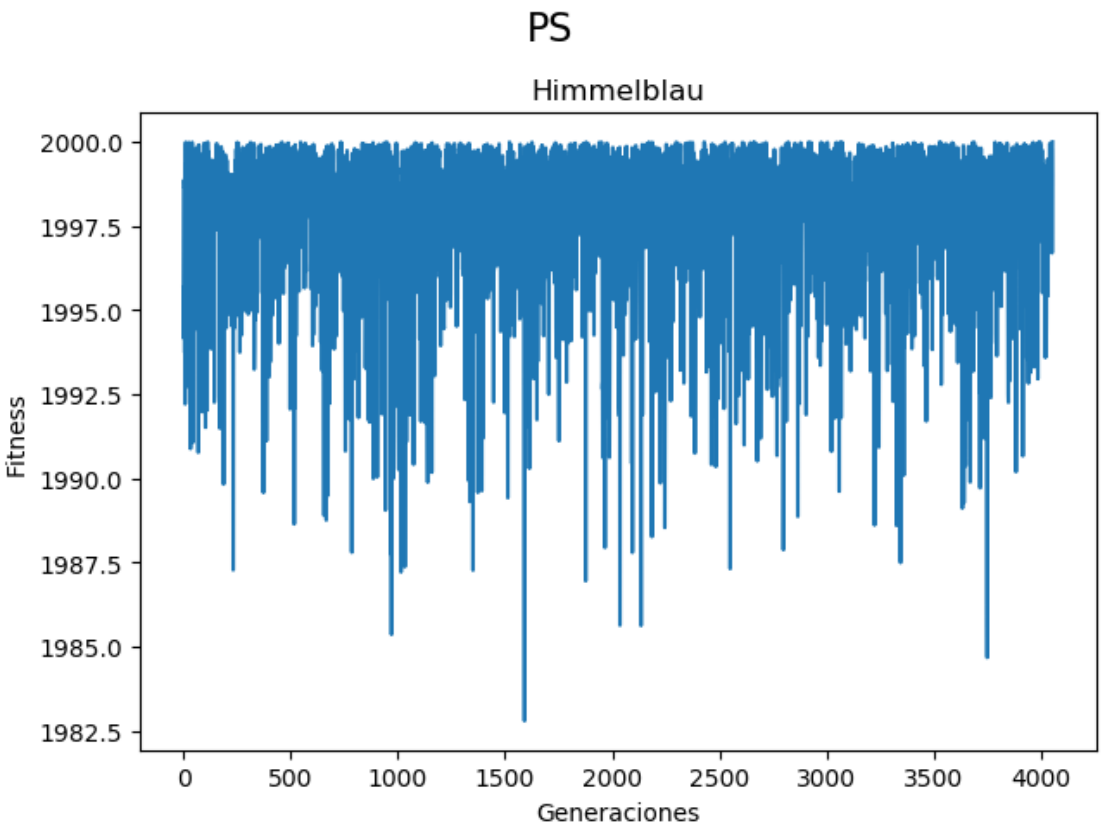

PS
Beale
+1.5e5
0
−2
−4
−6
−8
−10
Fitness
0
500
1000
1500
2000
2500
3000
Generaciones

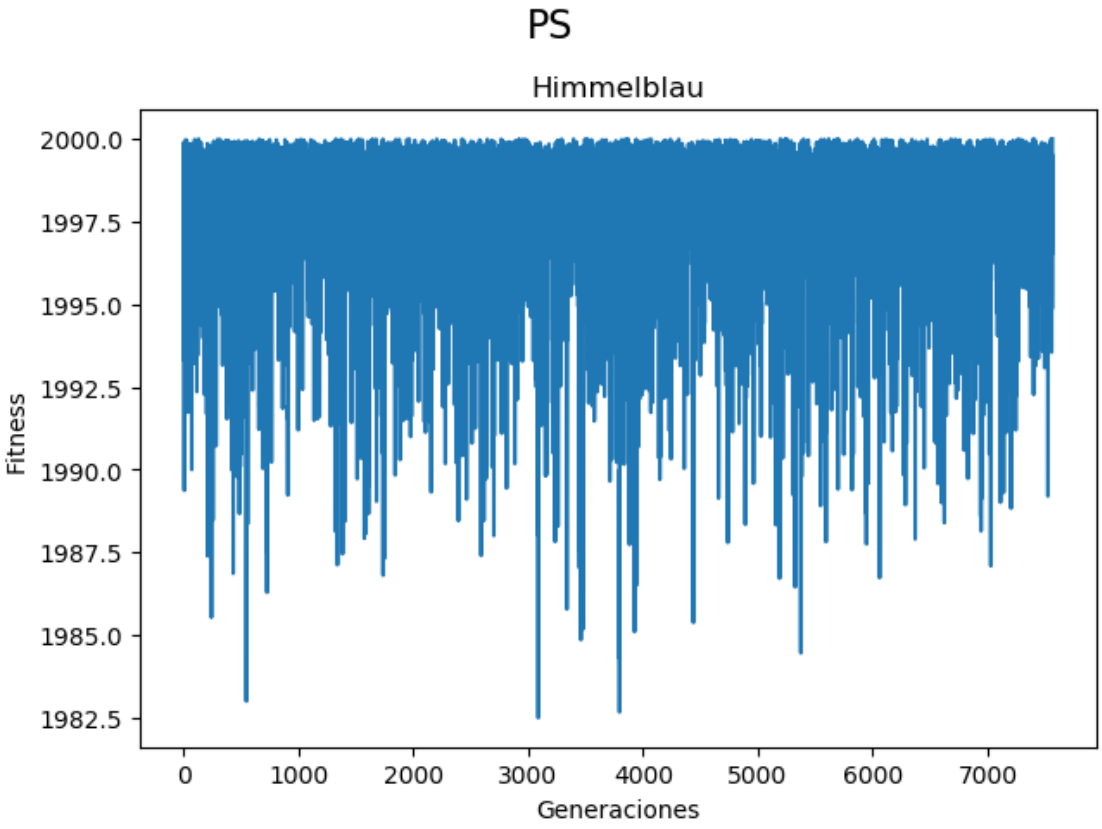


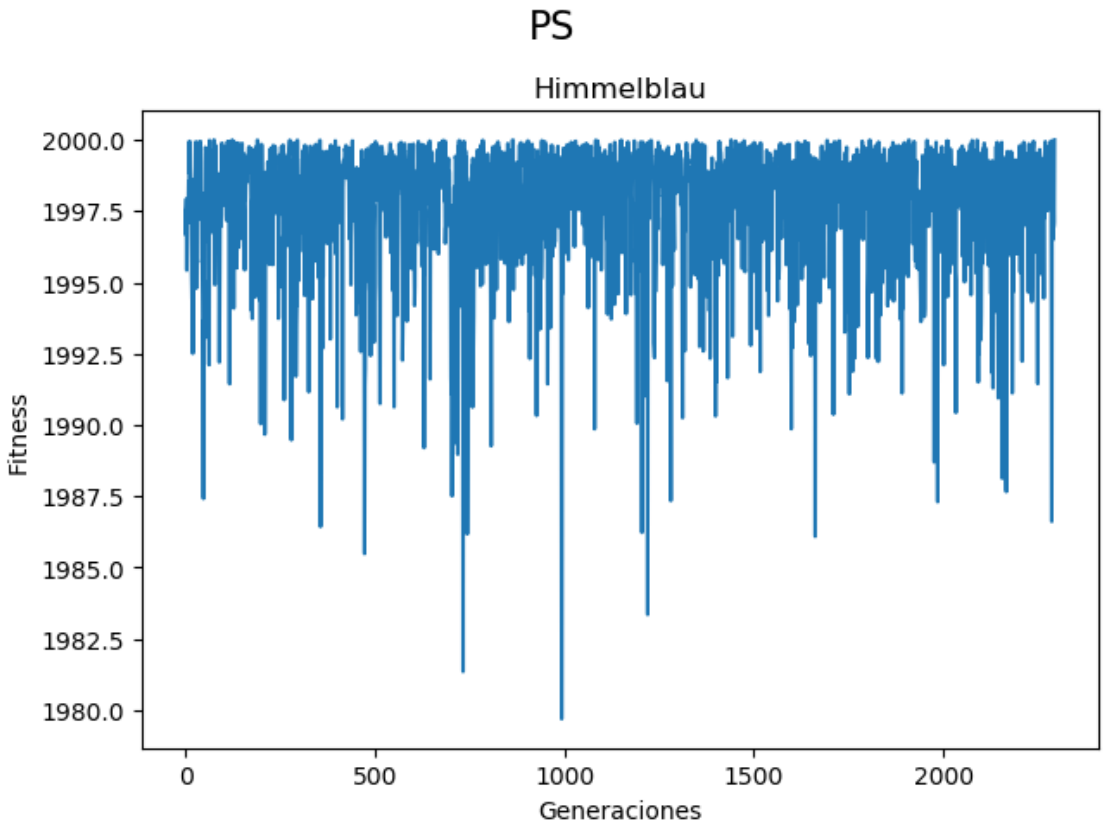
2.2.3. Himmelblau

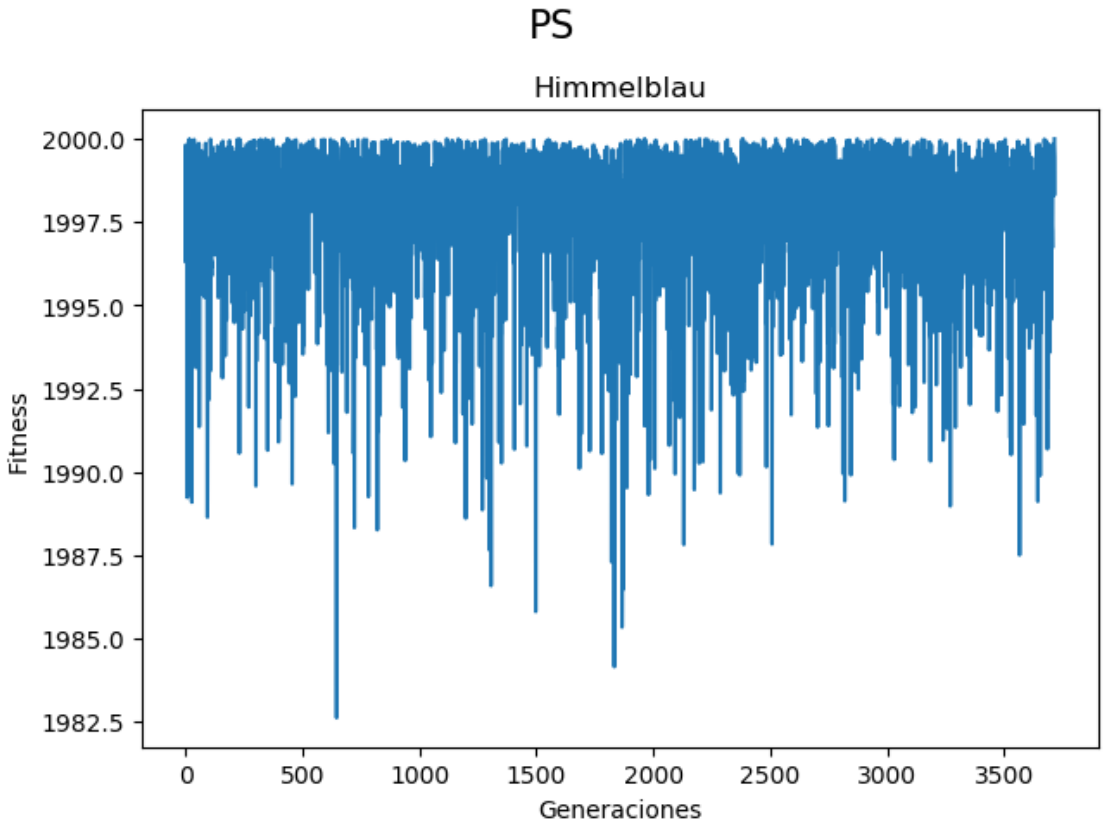
Run: 0
Selection method: PS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[3.58702535 -1.85506329]]
after: 4053 generations.
Run: 1
Selection method: PS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[3.00291445 2.00002289]]
after: 7564 generations.
Run: 2
Selection method: PS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[3.58099809 -1.85231668]]
after: 2294 generations.
Run: 3
Selection method: PS

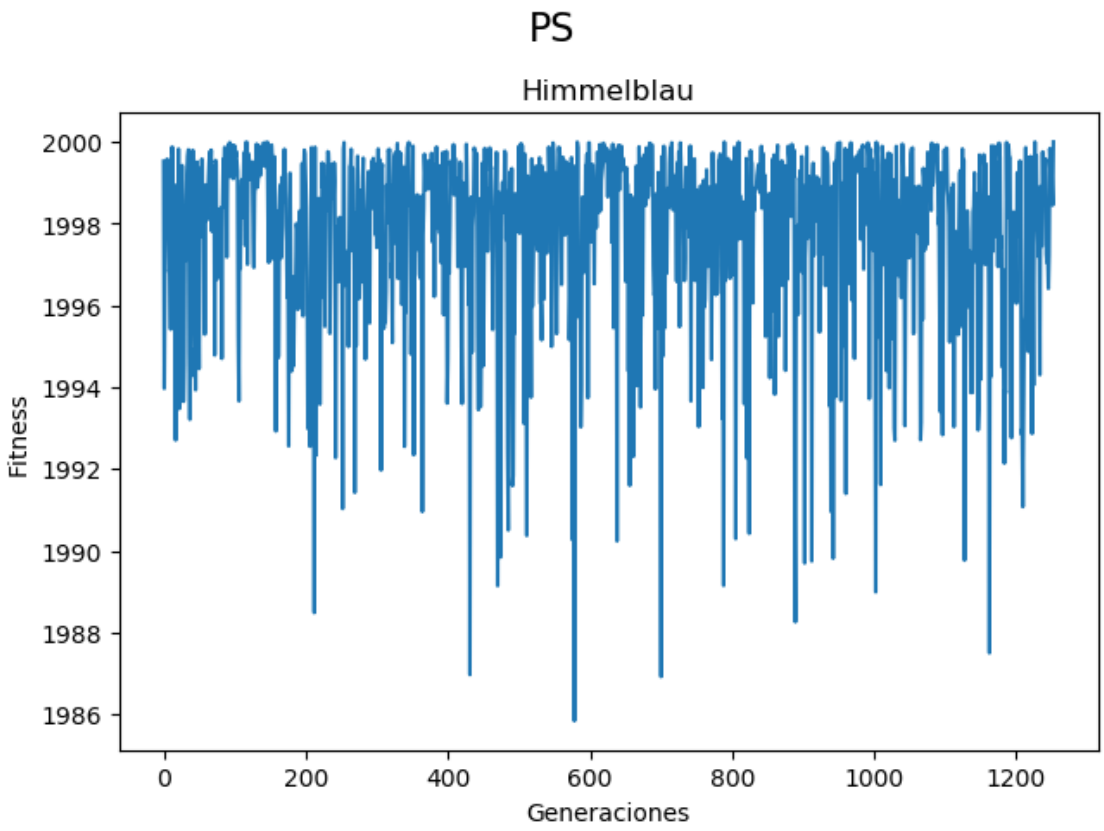
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[-2.80737158 3.12948707]]
after: 3716 generations.
Run: 4
Selection method: PS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000
Global minimum found: [[3.58488911 -1.84758642]]
after: 1253 generations.


PS
Himmelblau
Fitness
Generaciones
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
0
500
1000
1500
2000
2500
3000
3500
4000


PS
Himmelblau
Fitness
Generaciones
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
0
1000
2000
3000
4000
5000
6000
7000


PS
Himmelblau
Fitness
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
1980.0
0
500
1000
1500
2000
Generaciones

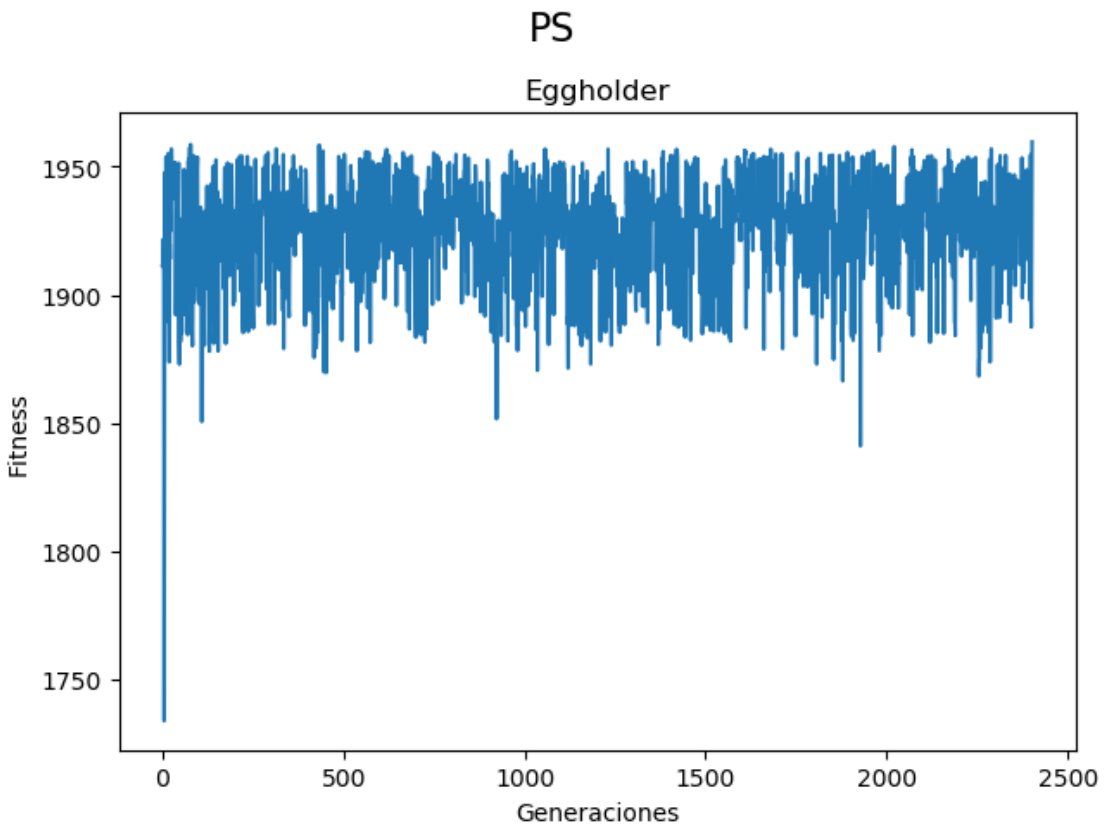

PS
Himmelblau
Fitness
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
0
500
1000
1500
2000
2500
3000
3500
Generaciones

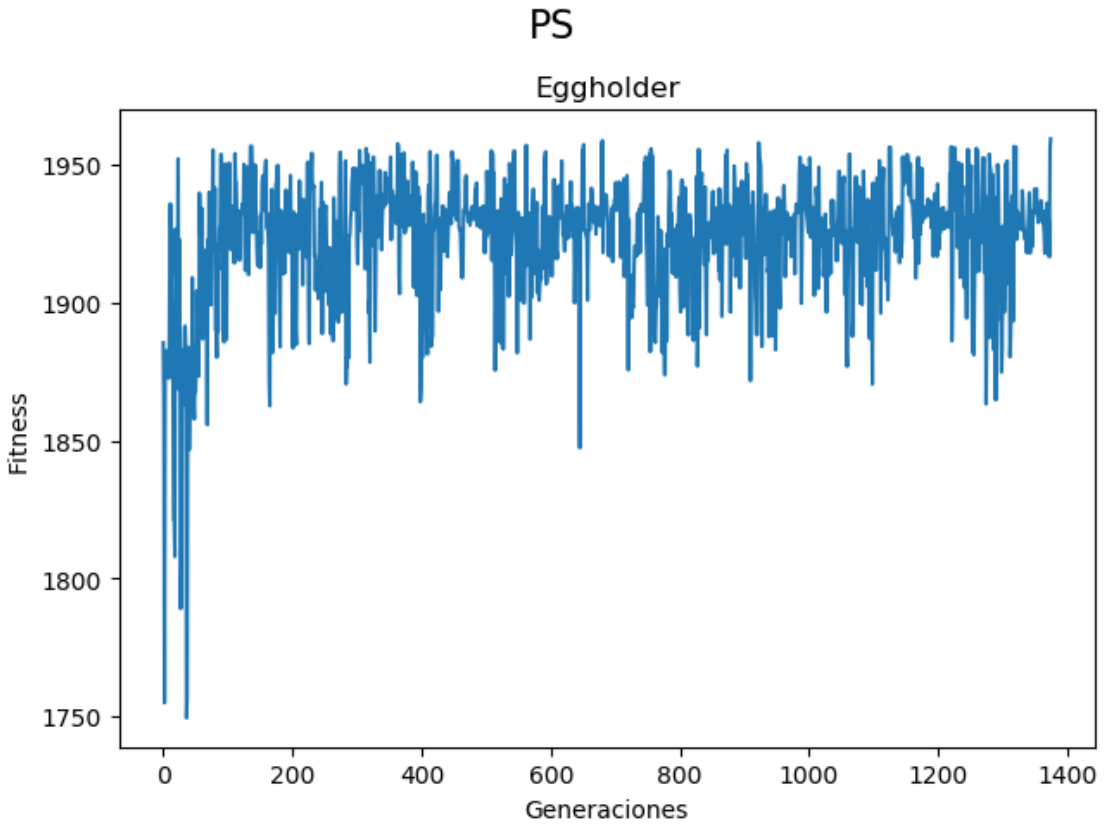


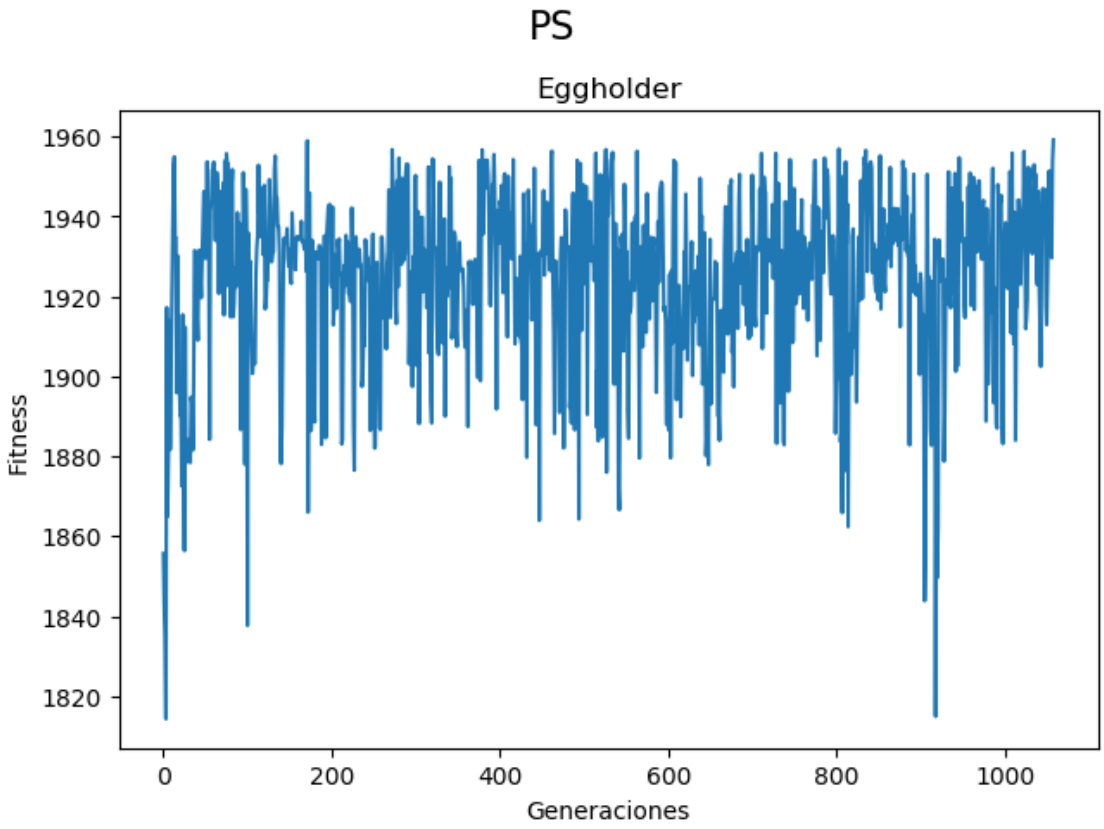
2.2.4. Eggholder

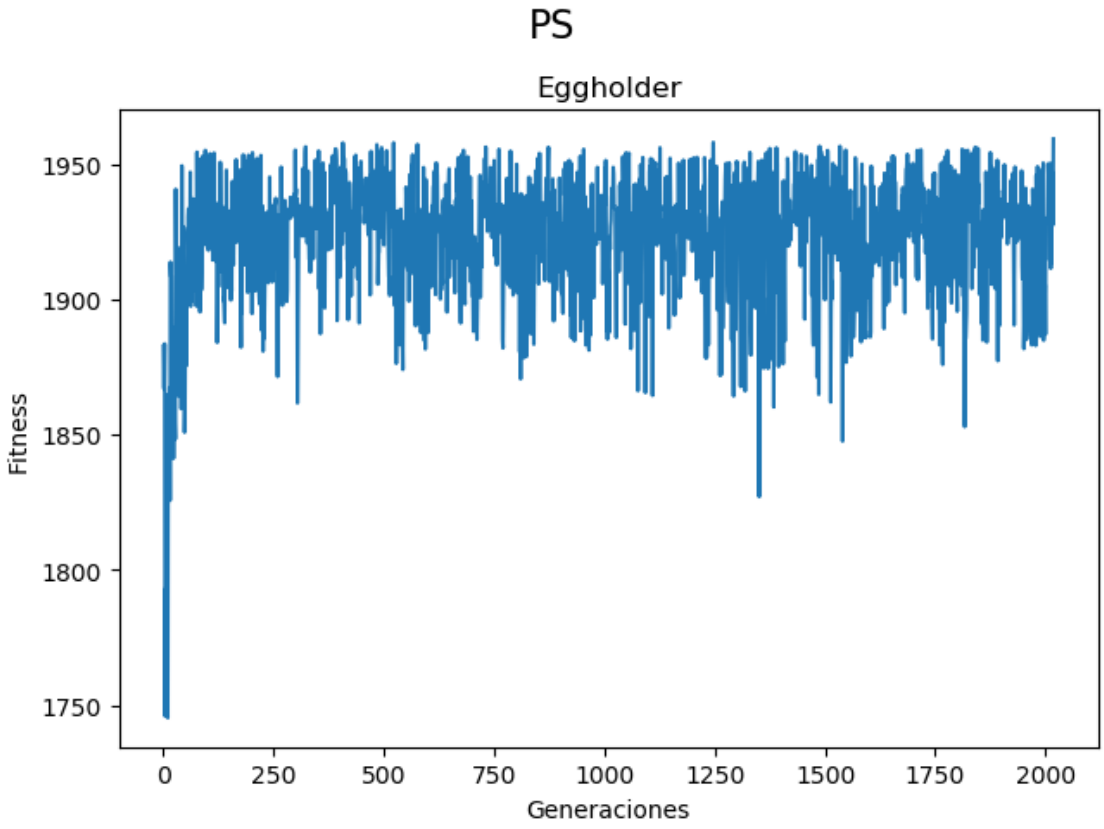
Run: 0
Selection method: PS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000
Global minimum found: [[511.99938965 404.16521574]]
after: 2403 generations.
Run: 1
Selection method: PS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000
Global minimum found: [[511.94348144 404.37273528]]
after: 1375 generations.
Run: 2
Selection method: PS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000
Global minimum found: [[511.85949706 404.10015226]]
after: 1059 generations.
Run: 3
Selection method: PS

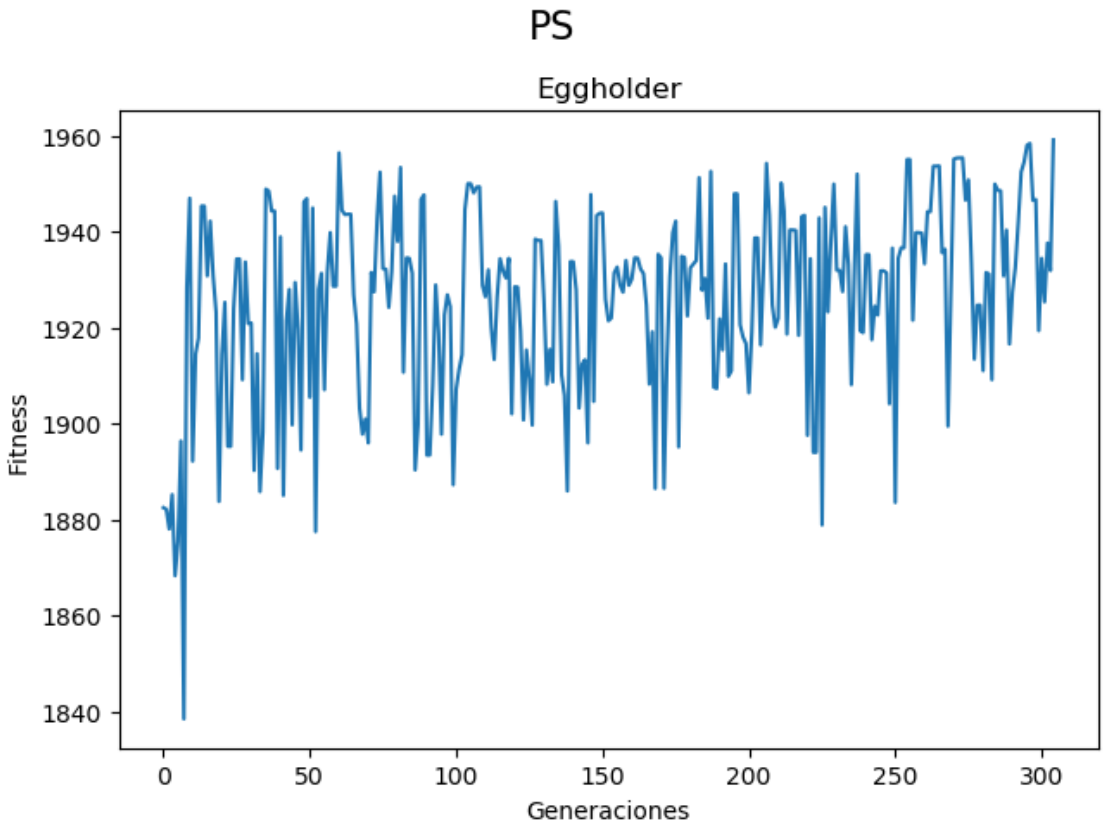
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000
Global minimum found: [[511.96417236 403.93395352]]
after: 2019 generations.
Run: 4
Selection method: PS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000
Global minimum found: [[511.86413573 404.06493497]]
after: 305 generations.


PS
Eggholder
1950
1900
1850
1800
1750
Fitness
0
500
1000
1500
2000
2500
Generaciones


PS
Eggholder
1950
1900
1850
1800
1750
Fitness
0
200
400
600
800
1000
1200
1400
Generaciones


PS
Eggholder
Fitness
Generaciones
1960
1940
1920
1900
1880
1860
1840
1820
0
200
400
600
800
1000


PS
Eggholder
Fitness
Generaciones
1950
1900
1850
1800
1750
0
250
500
750
1000
1250
1500
1750
2000

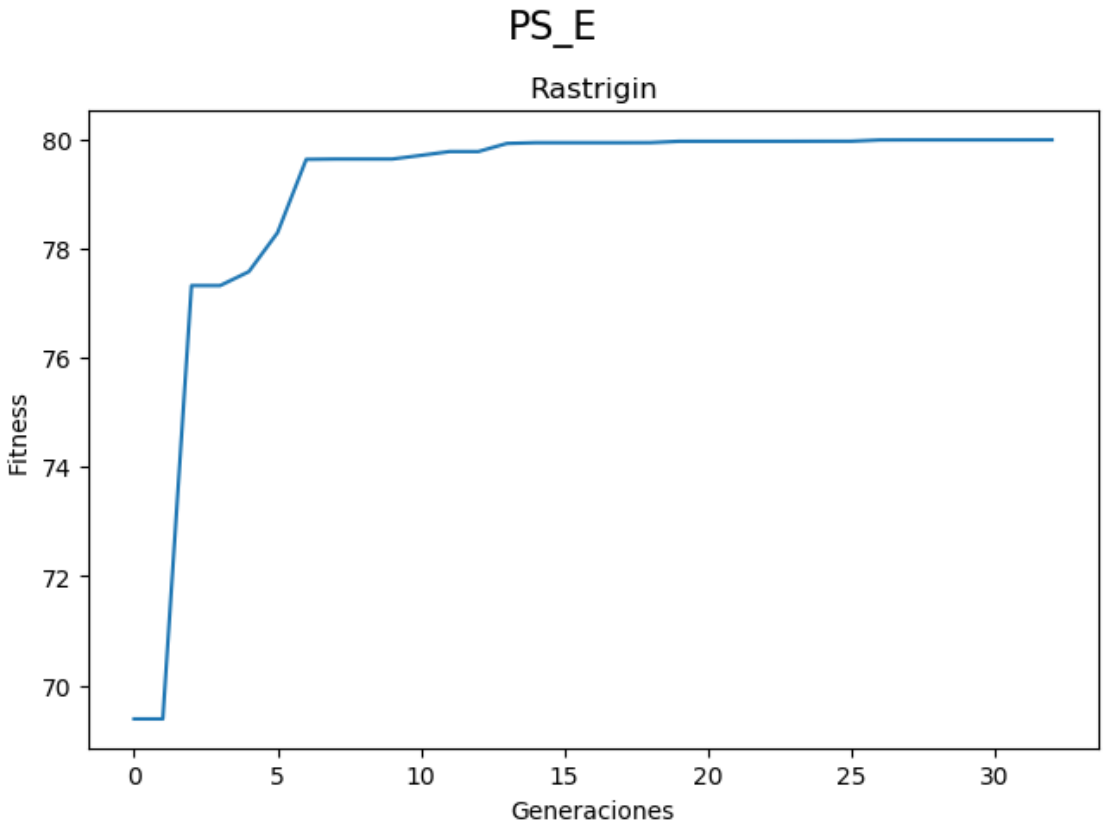


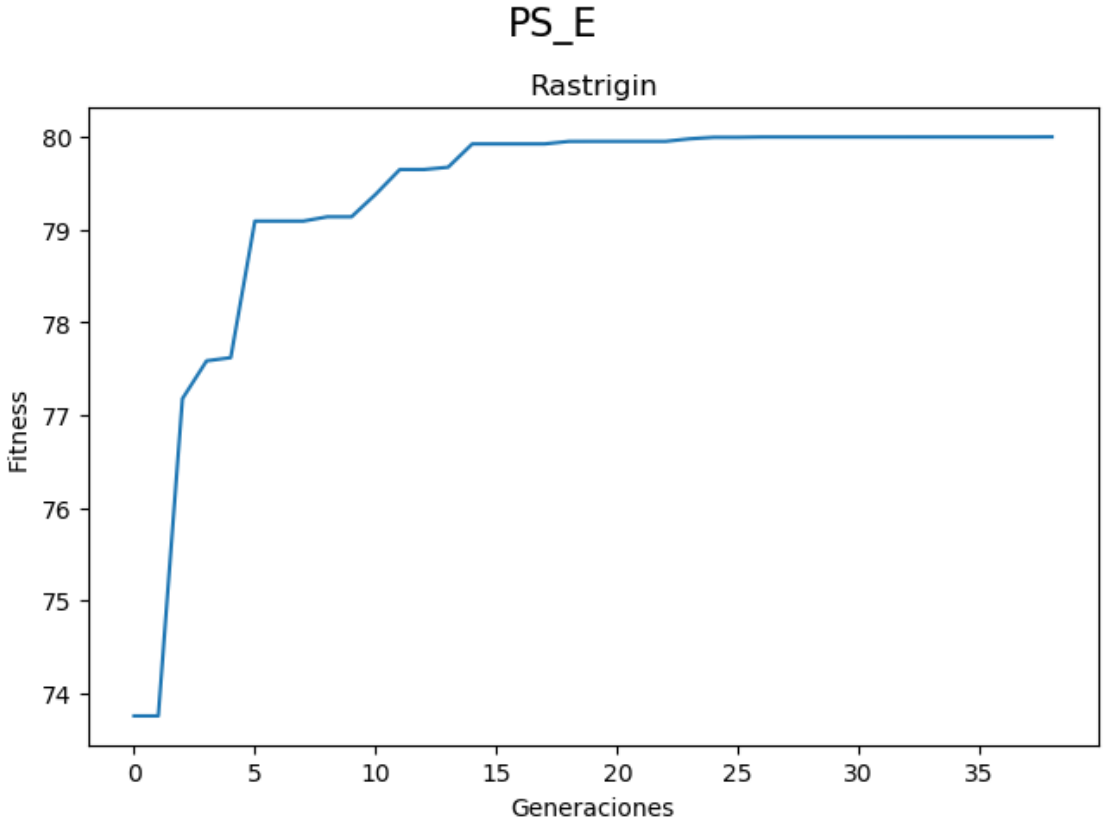
2.3. Selección proporcional, con elitismo

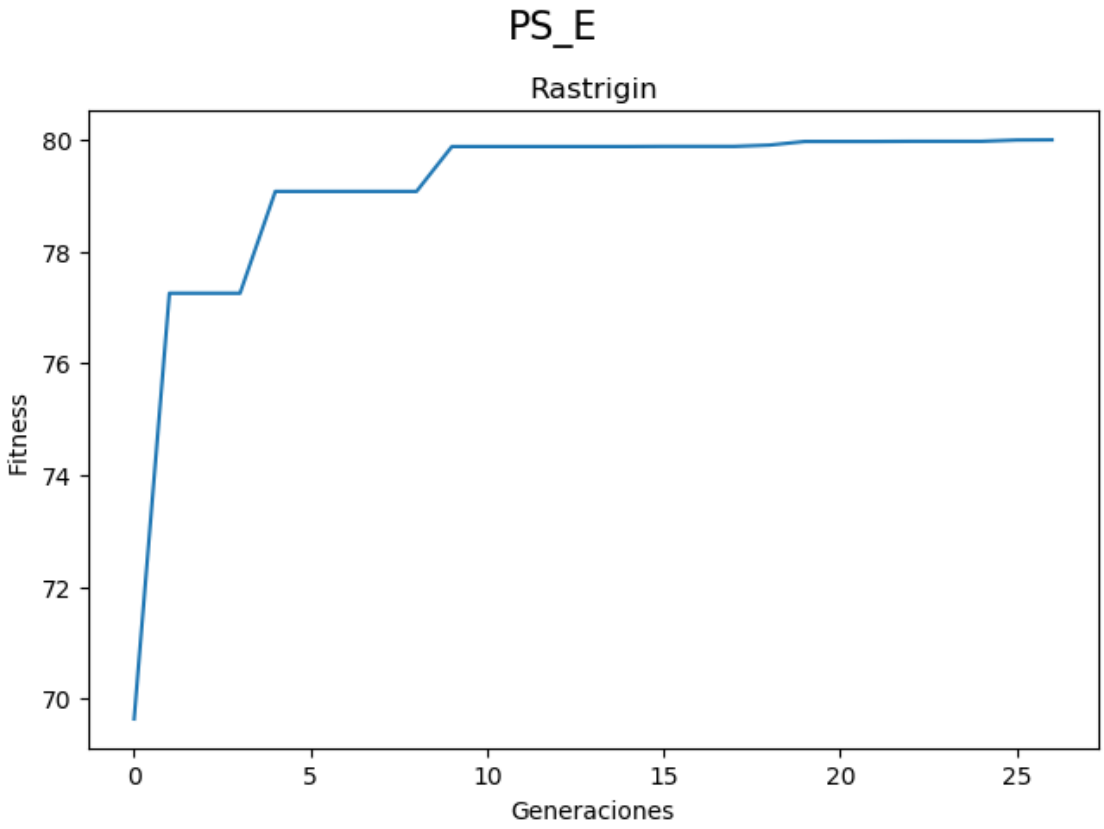
2.3.1. Rastrigin

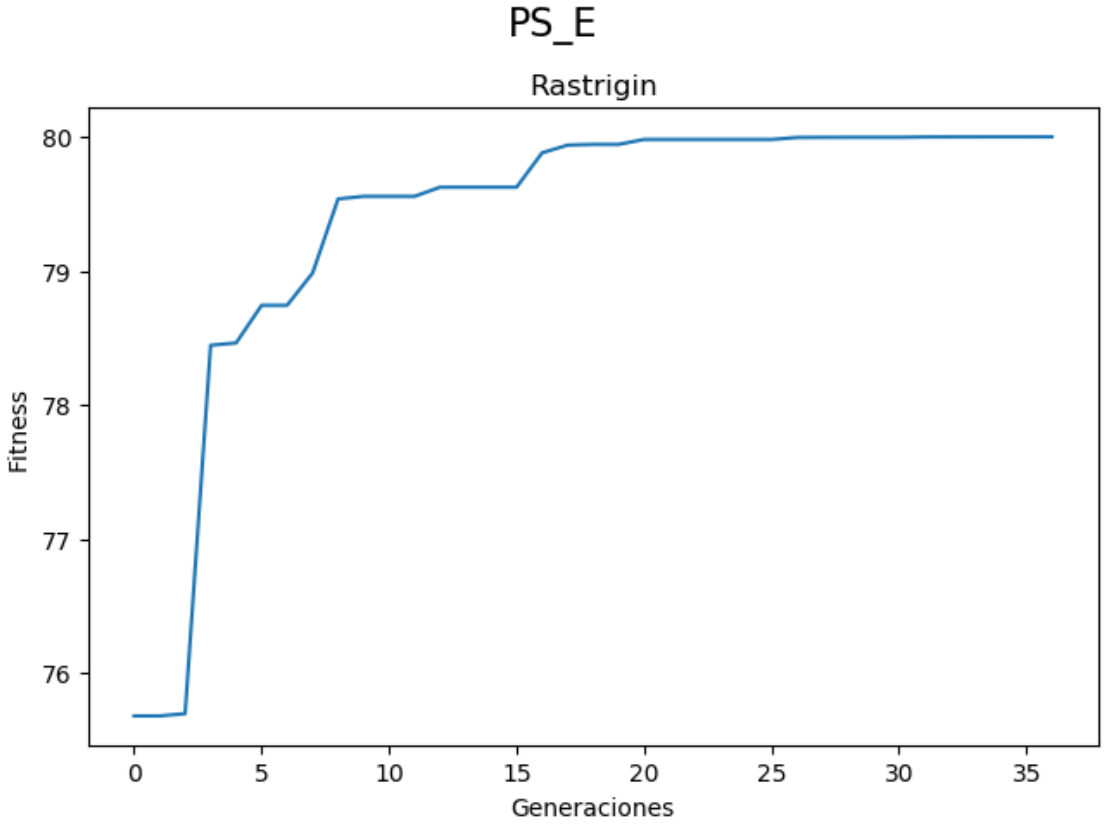
Run: 0
Selection method: PS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[0.00128907 -0.00097657]]
after: 33 generations.
Run: 1
Selection method: PS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[8.20318759e-04 -3.90627980e-05]]
after: 39 generations.
Run: 2
Selection method: PS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.90627980e-05 1.60157472e-03]]
after: 27 generations.

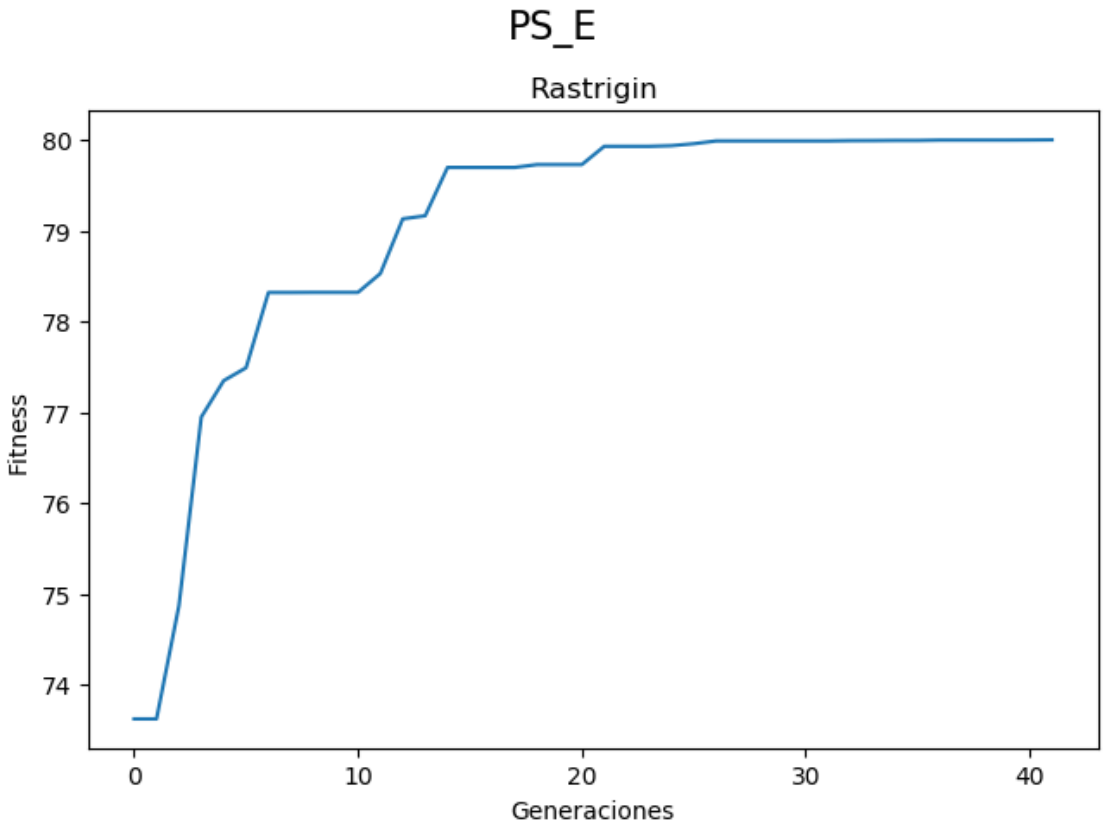
Run: 3
Selection method: PS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[0.0019922 -0.00082032]]
after: 37 generations.
Run: 4
Selection method: PS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[0.00082032 -0.00121095]]
after: 42 generations.


PS_E
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
5
10
15
20
25
30


PS_E
Rastrigin
Fitness
Generaciones
80
79
78
77
76
75
74
0
5
10
15
20
25
30
35


PS_E
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
5
10
15
20
25

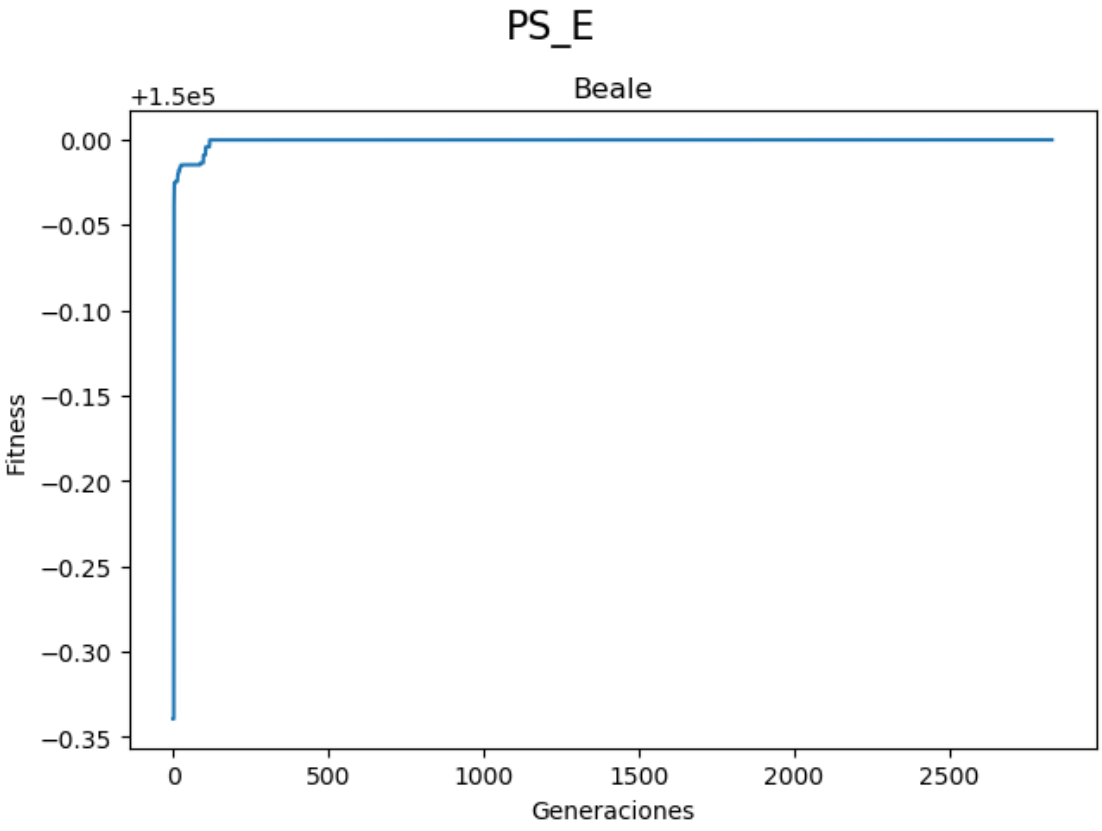

PS_E
Rastrigin
Fitness
Generaciones
80
79
78
77
76
0
5
10
15
20
25
30
35

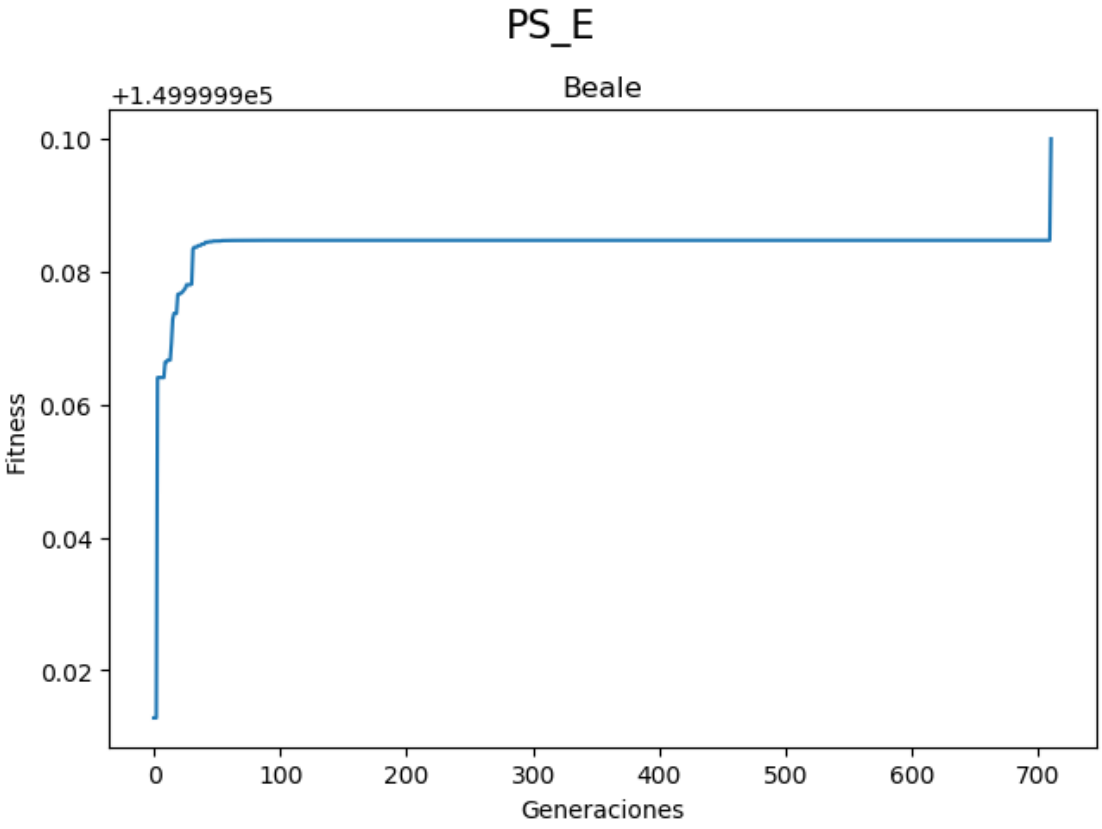


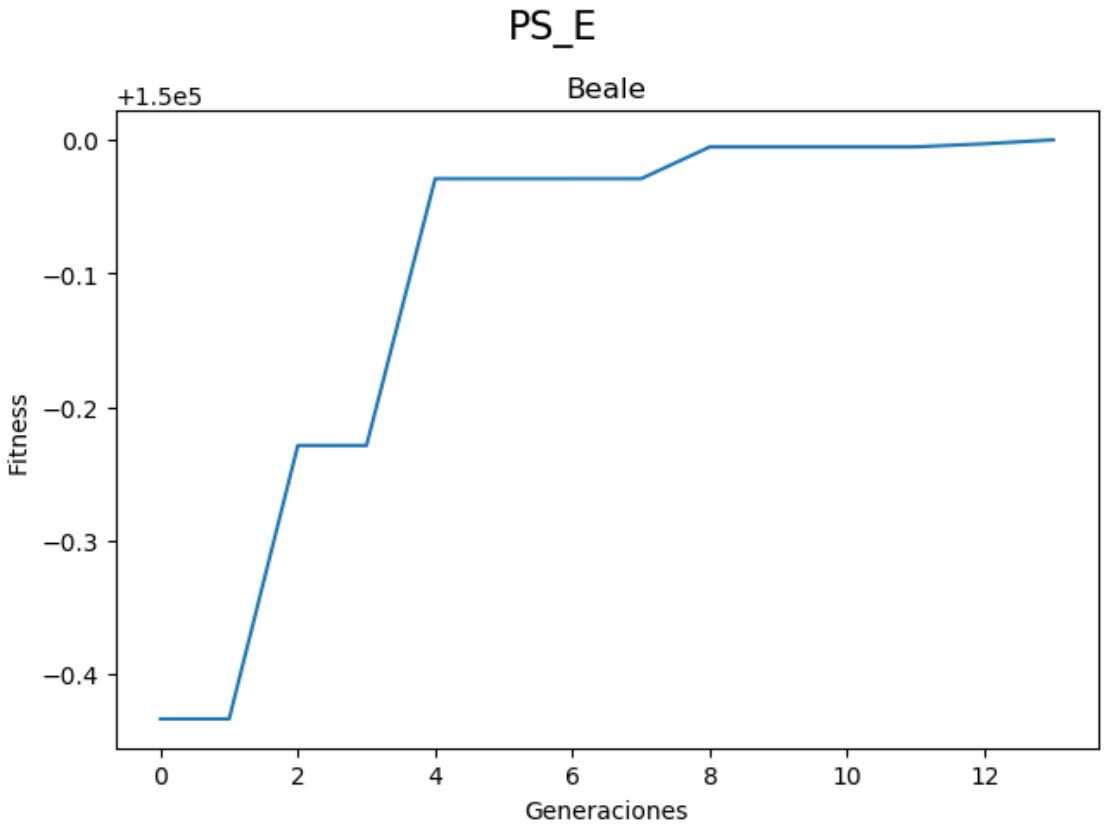
2.3.2. Beale

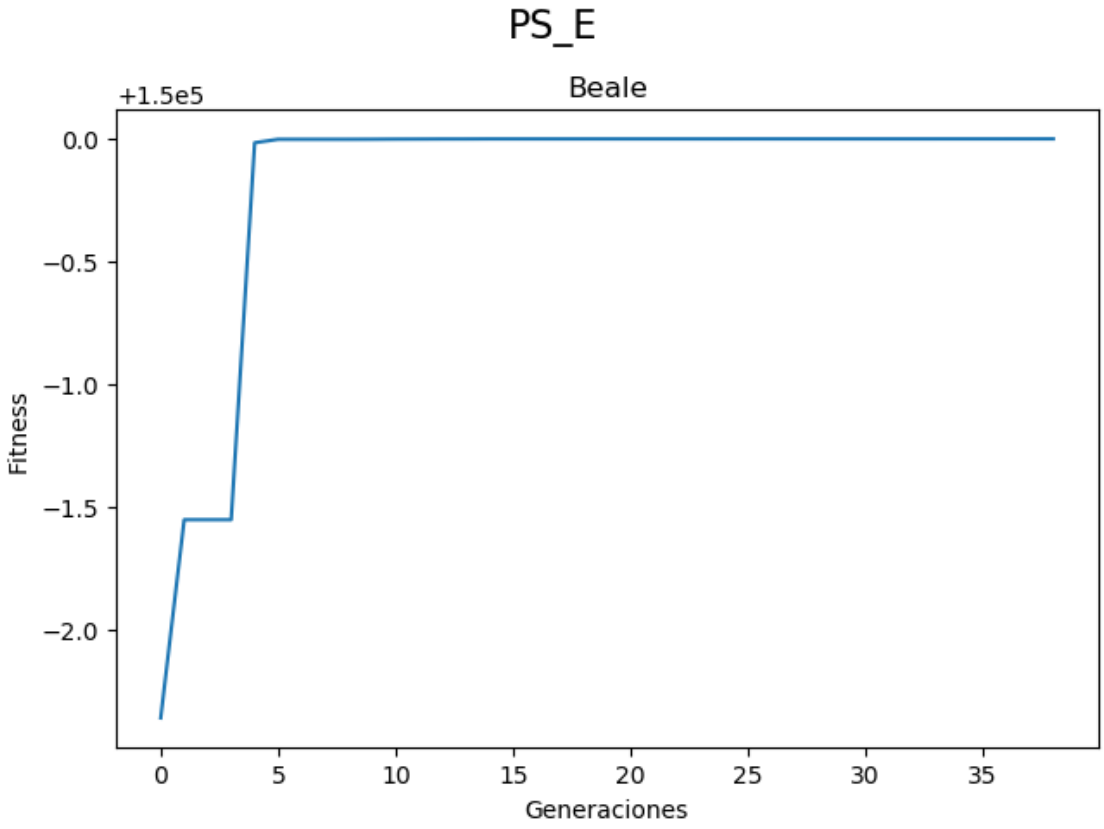
Run: 0
Selection method: PS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.0250742 0.50643926]]
after: 2831 generations.
Run: 1
Selection method: PS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[2.98833838 0.49662015]]
after: 711 generations.
Run: 2
Selection method: PS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.01807036 0.50362399]]
after: 14 generations.
Run: 3
Selection method: PS_E

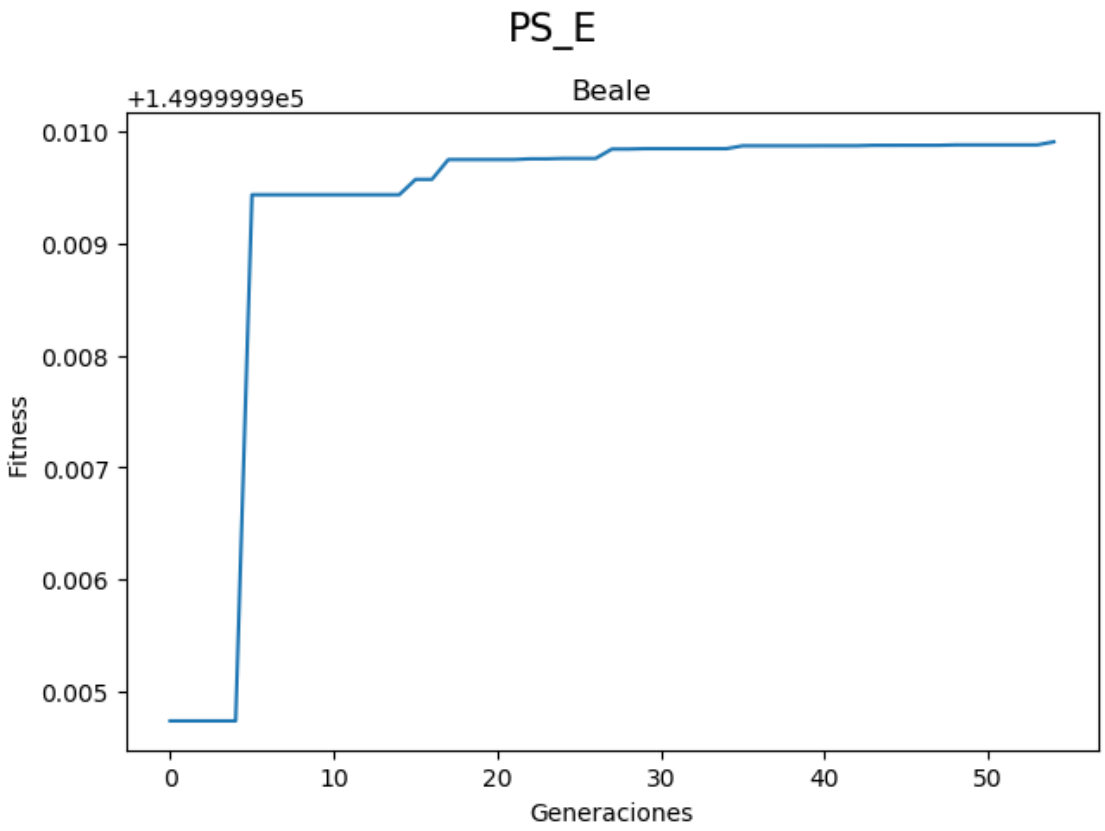
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.01470577 0.50293734]]
after: 39 generations.
Run: 4
Selection method: PS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.02411289 0.50588994]]
after: 55 generations.


PS_E
Beale
+1.5e5
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
Fitness
0
500
1000
1500
2000
2500
Generaciones


PS_E
Beale
+1.499999e5
0.10
0.08
0.06
0.04
0.02
Fitness
0
100
200
300
400
500
600
700
Generaciones


PS_E
Beale
+1.5e5
0.0
−0.1
−0.2
−0.3
−0.4
Fitness
0
2
4
6
8
10
12
Generaciones

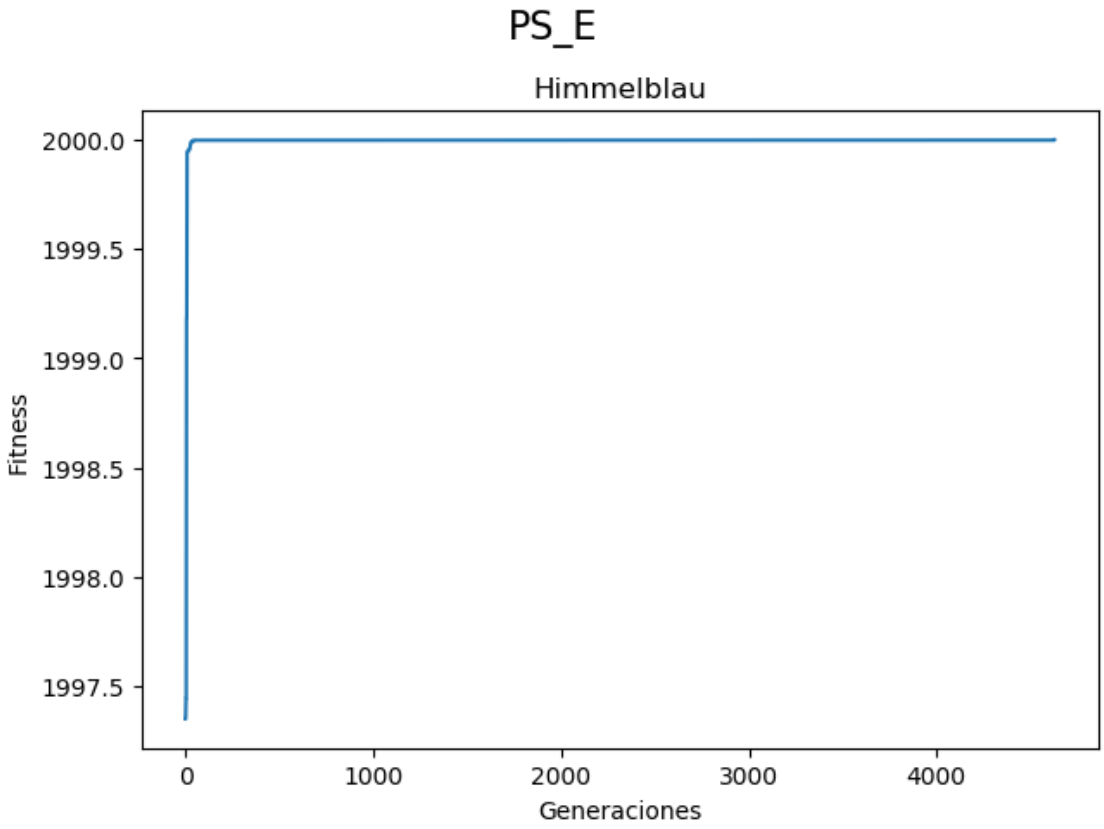

PS_E
Beale
+1.5e5
0.0
−0.5
−1.0
−1.5
−2.0
Fitness
0
5
10
15
20
25
30
35
Generaciones

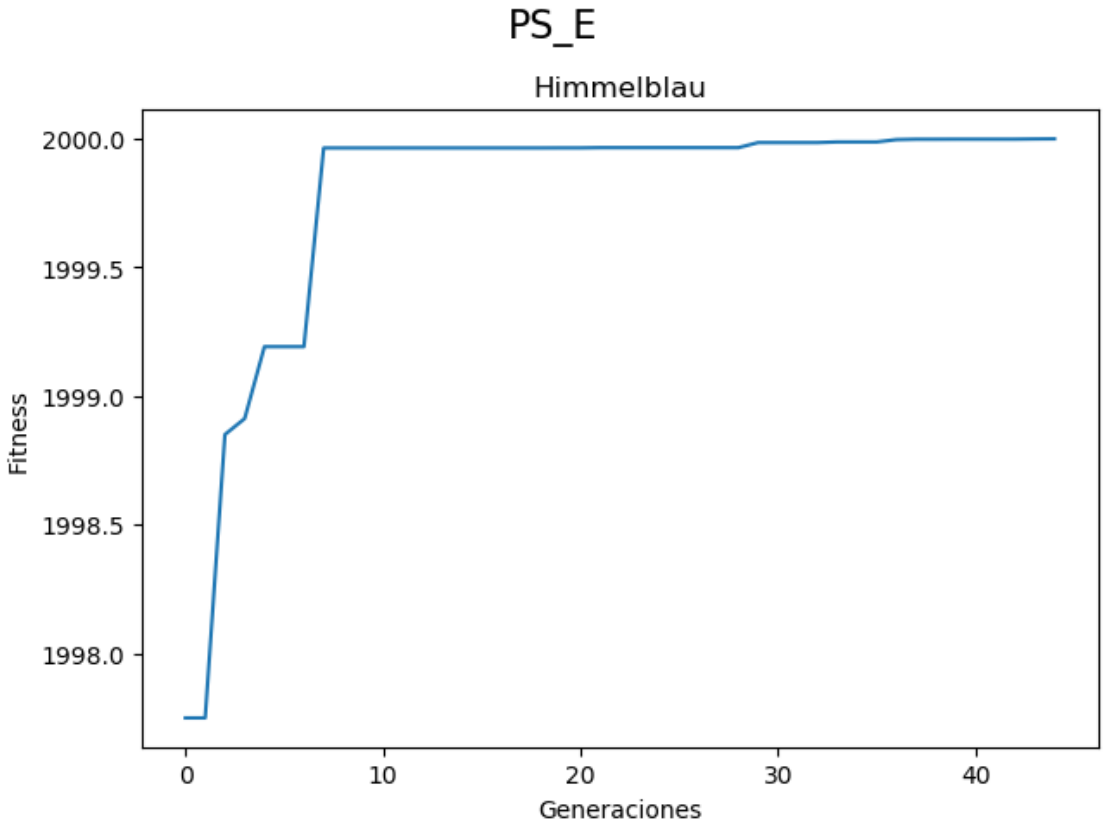


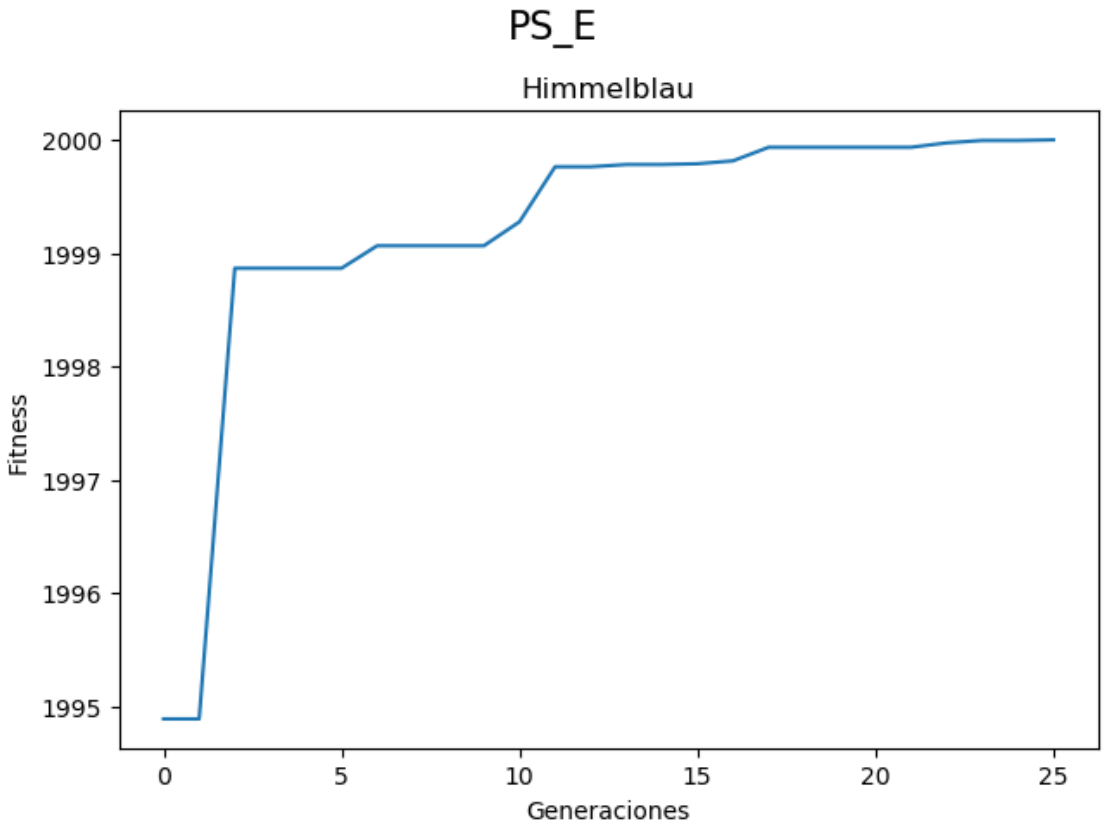
2.3.3. Himmelblau

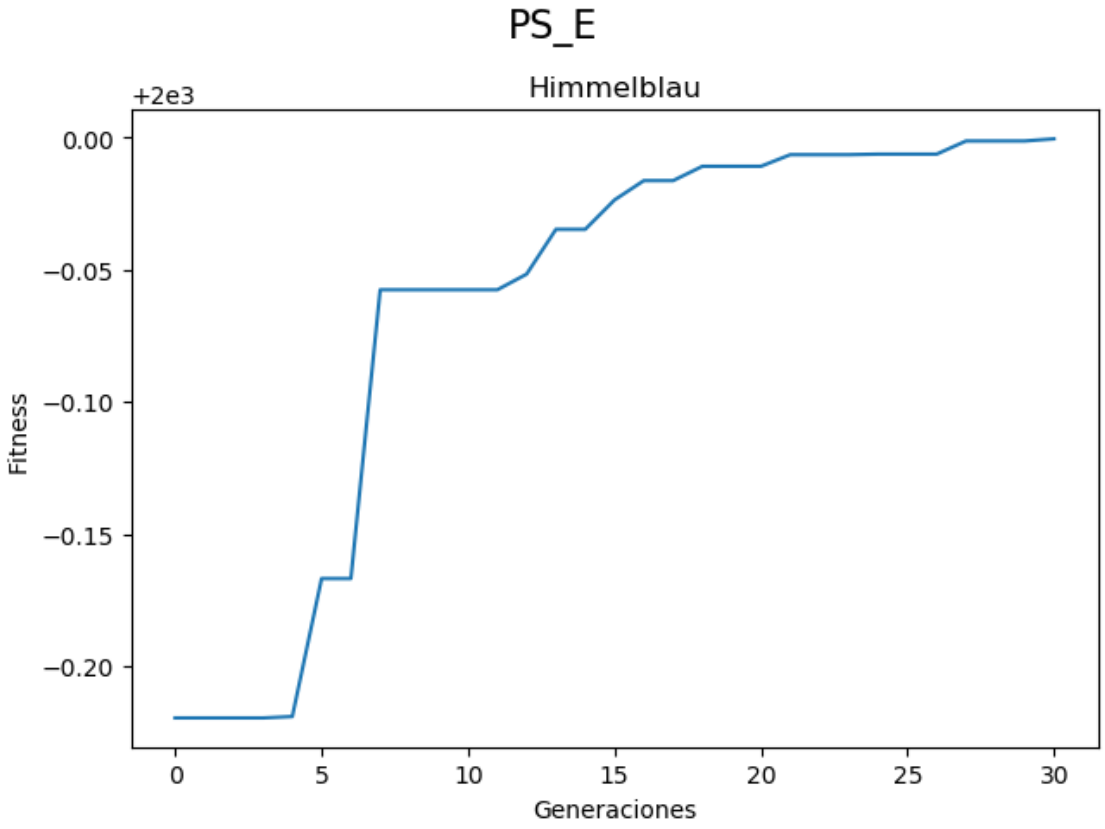
Run: 0
Selection method: PS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[-2.8056168 3.13124185]]
after: 4628 generations.
Run: 1
Selection method: PS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[2.99551388 2.00452427]]
after: 45 generations.
Run: 2
Selection method: PS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.00177003 1.99170678]]
after: 26 generations.
Run: 3
Selection method: PS_E

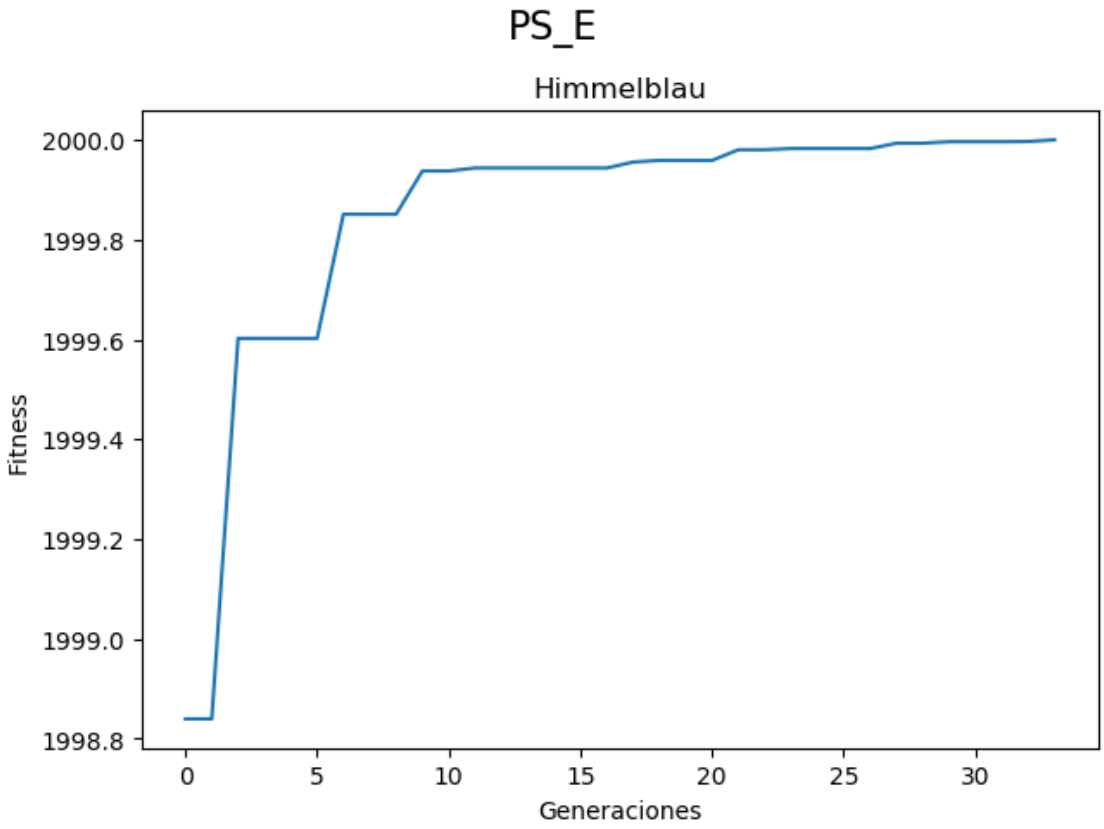
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.00291445 1.99552151]]
after: 31 generations.
Run: 4
Selection method: PS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'elitism': 0.2
Global minimum found: [[3.58534687 -1.85040932]]
after: 34 generations.


PS_E
Himmelblau
2000.0
1999.5
1999.0
1998.5
1998.0
1997.5
Fitness
0
1000
2000
3000
4000
Generaciones


PS_E
Himmelblau
2000.0
1999.5
1999.0
1998.5
1998.0
Fitness
0
10
20
30
40
Generaciones


PS_E
Himmelblau
2000
1999
1998
1997
1996
1995
Fitness
0
5
10
15
20
25
Generaciones

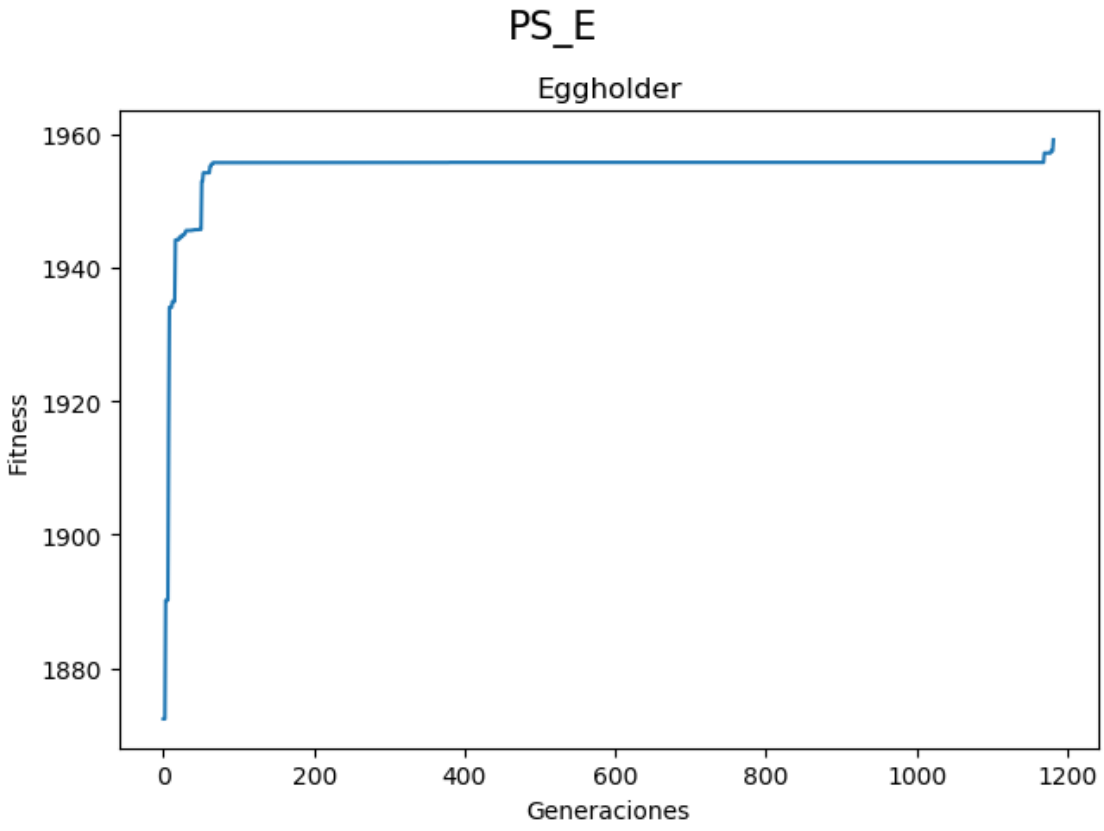

PS_E
Himmelblau
+2e3
0.00
−0.05
−0.10
−0.15
−0.20
Fitness
0
5
10
15
20
25
30
Generaciones

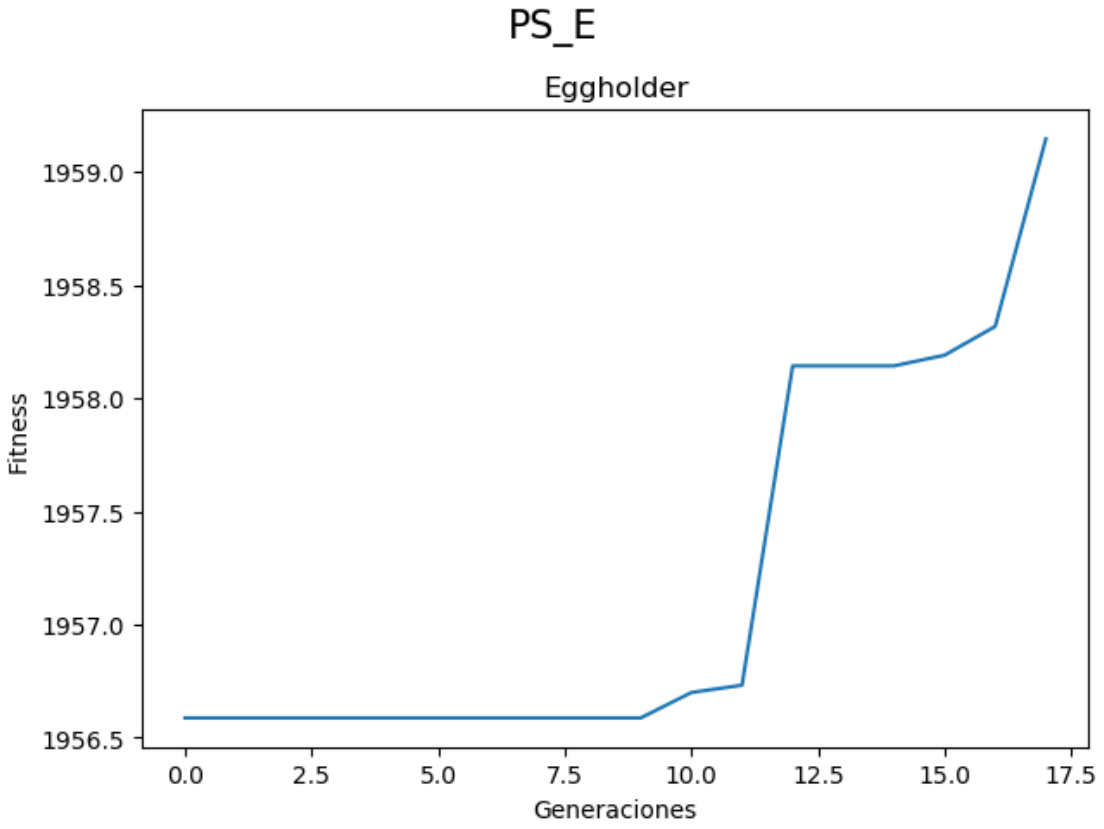


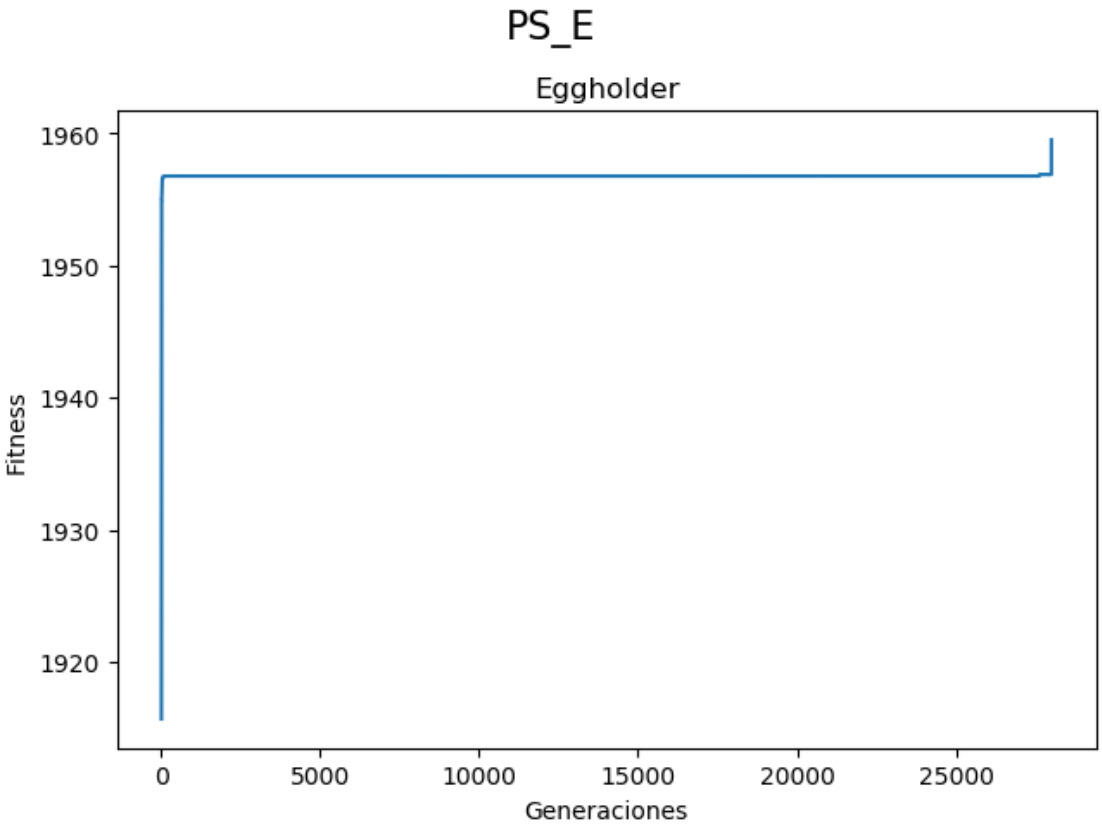
2.3.4. Eggholder

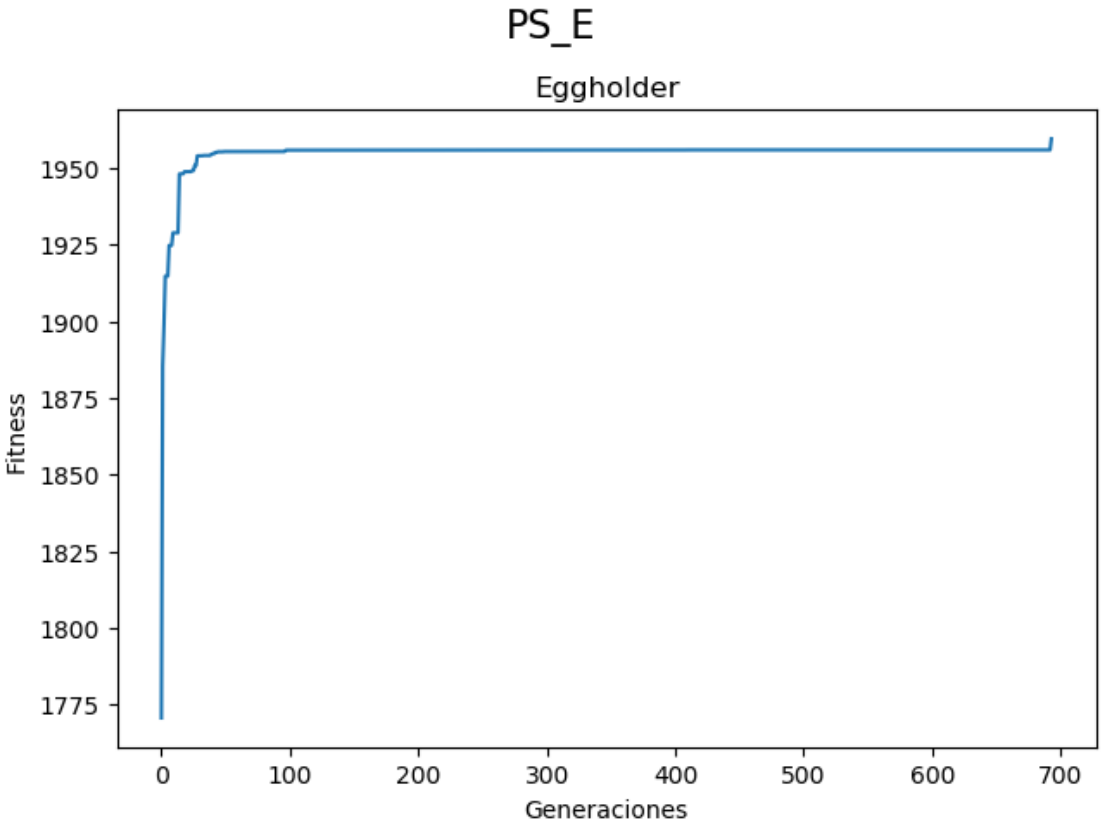
Run: 0
Selection method: PS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'elitism': 0.2
Global minimum found: [[511.93359375 404.53783539] [511.93359375 404.53783539]]
after: 1183 generations.
Run: 1
Selection method: PS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'elitism': 0.2
Global minimum found: [[511.95318603 403.64586757]]
after: 18 generations.
Run: 2
Selection method: PS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'elitism': 0.2
Global minimum found: [[511.97528076 404.40624359]]

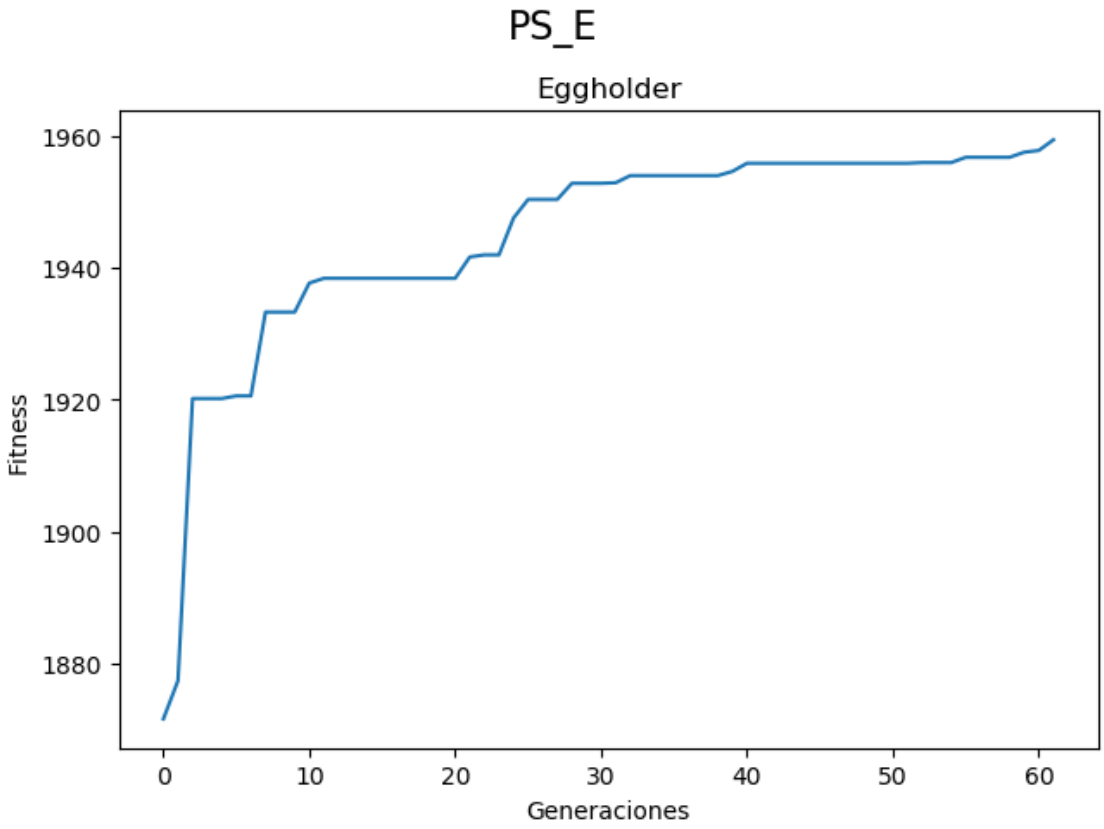
after: 28009 generations.
Run: 3
Selection method: PS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'elitism': 0.2
Global minimum found: [[512. 404.59838227]]
after: 694 generations.
Run: 4
Selection method: PS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'elitism': 0.2
Global minimum found: [[511.94763183 404.415643]]
after: 62 generations.


PS_E
Eggholder
1960
1940
1920
1900
1880
Fitness
0
200
400
600
800
1000
1200
Generaciones


PS_E
Eggholder
1959.0
1958.5
1958.0
1957.5
1957.0
1956.5
Fitness
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Generaciones


PS_E
Eggholder
1960
1950
1940
1930
1920
Fitness
0
5000
10000
15000
20000
25000
Generaciones


PS_E
Eggholder
1950
1925
1900
1875
1850
1825
1800
1775
Fitness
0
100
200
300
400
500
600
700
Generaciones

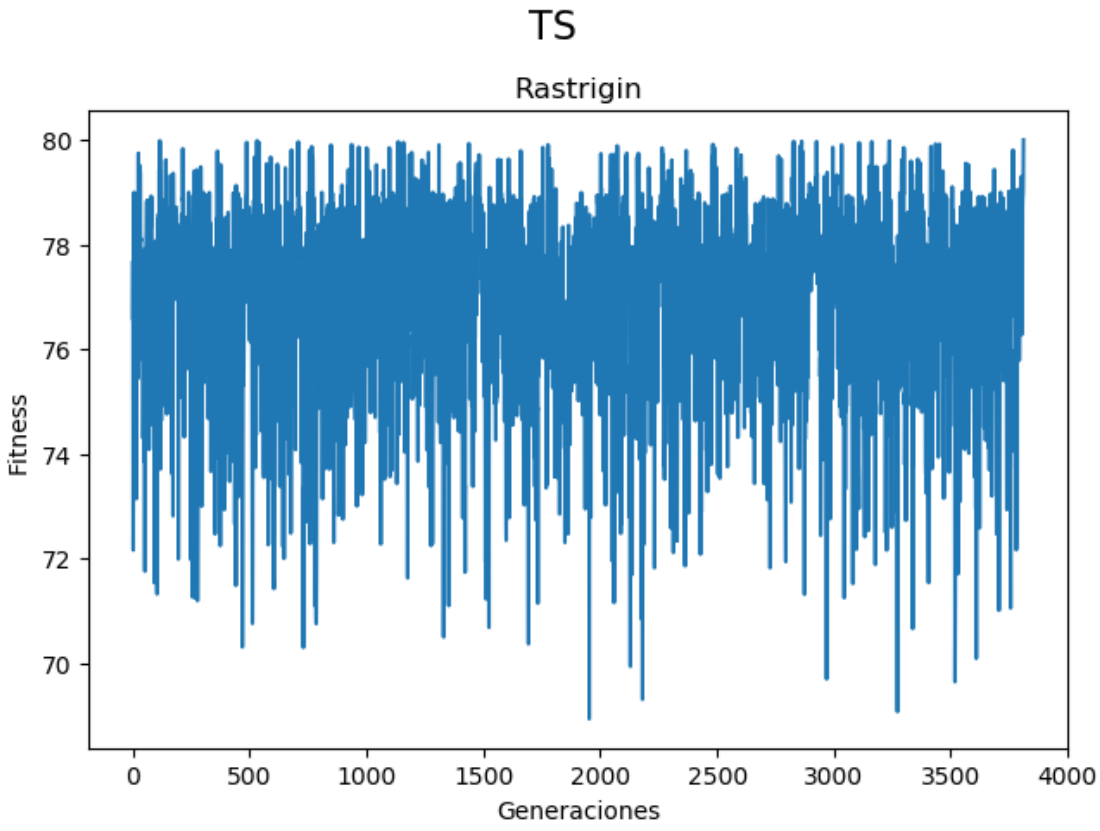


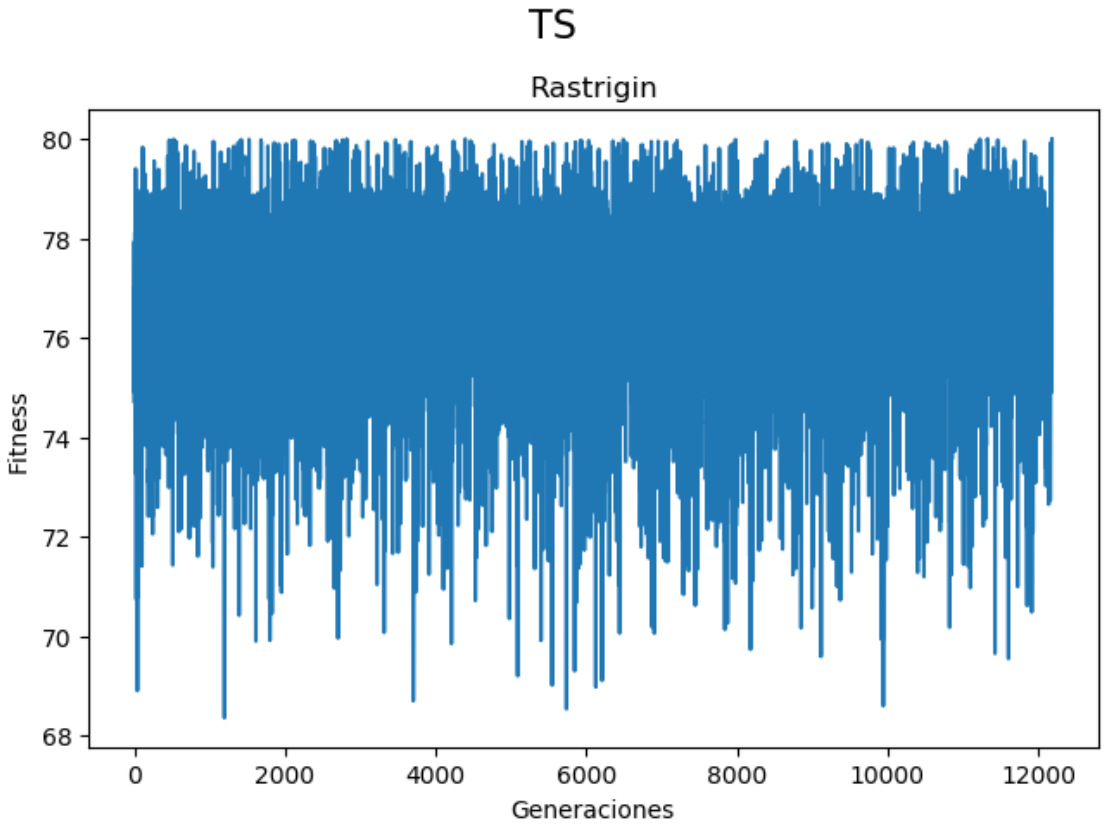
2.4. Selección por torneo, sin elitismo

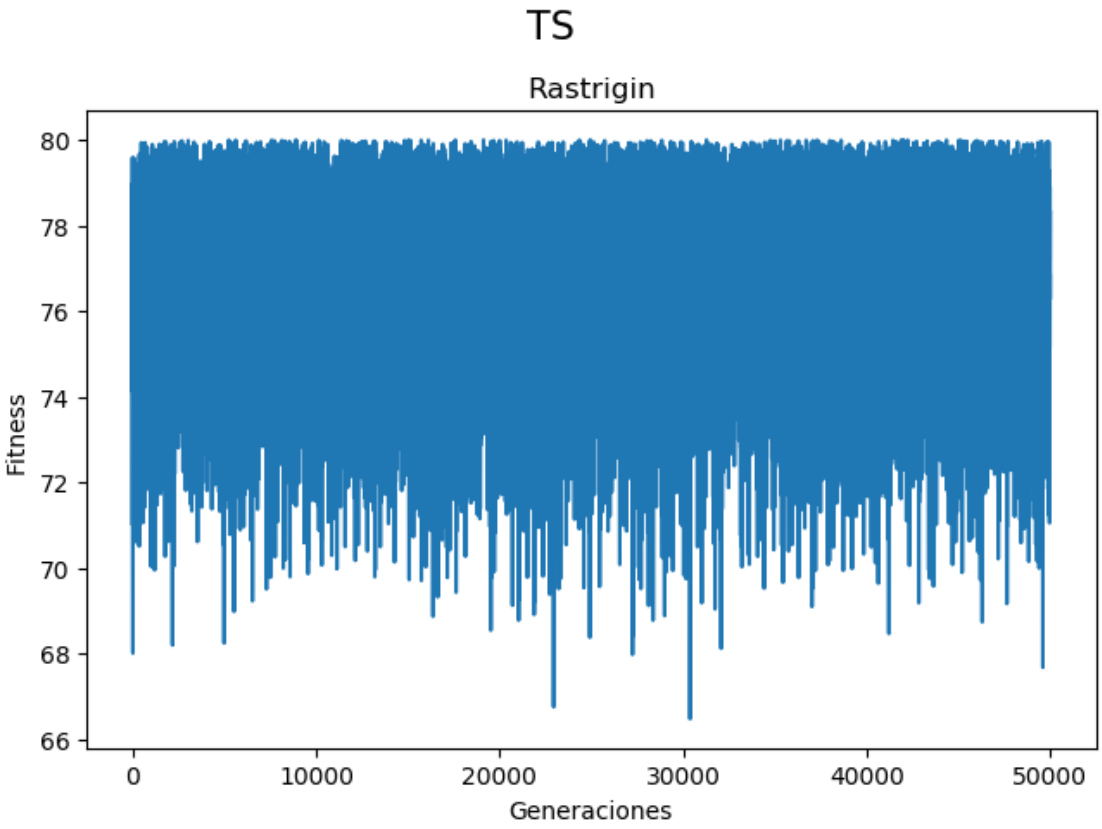
2.4.1. Rastrigin

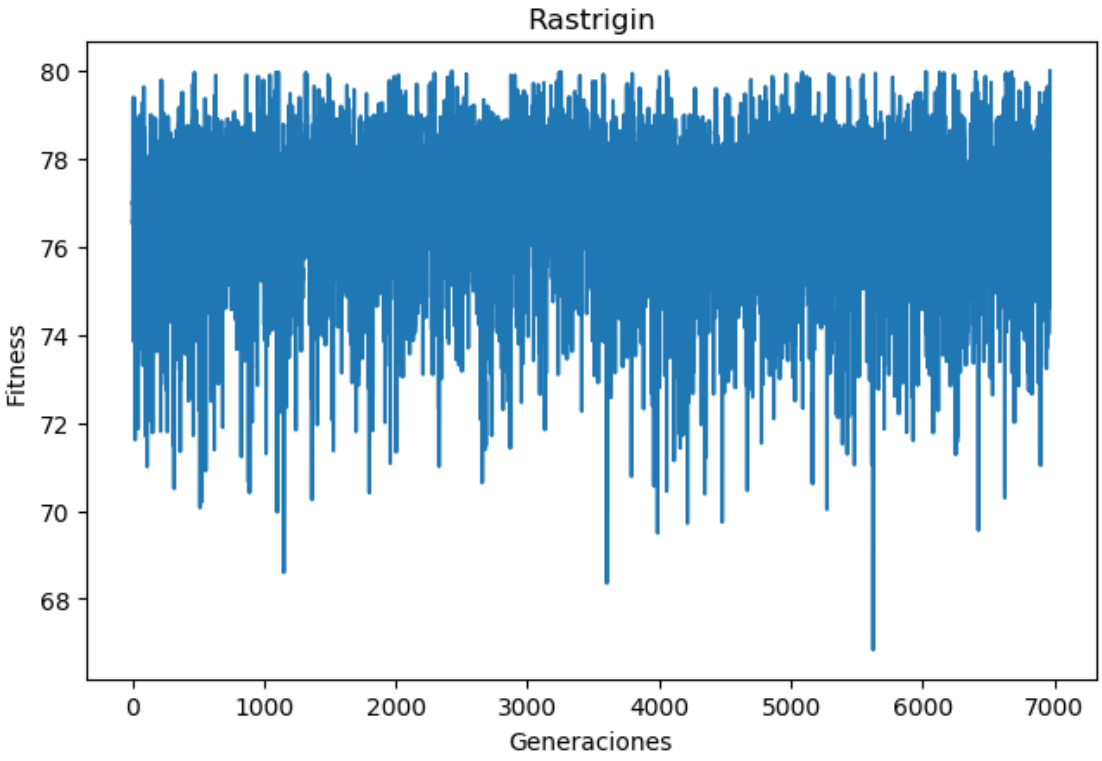
Run: 0
Selection method: TS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[-0.0013672 -0.00152345]]
after: 3812 generations.
Run: 1
Selection method: TS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[0.00152345 0.00082032]]
after: 12186 generations.
Run: 2
Selection method: TS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tour-

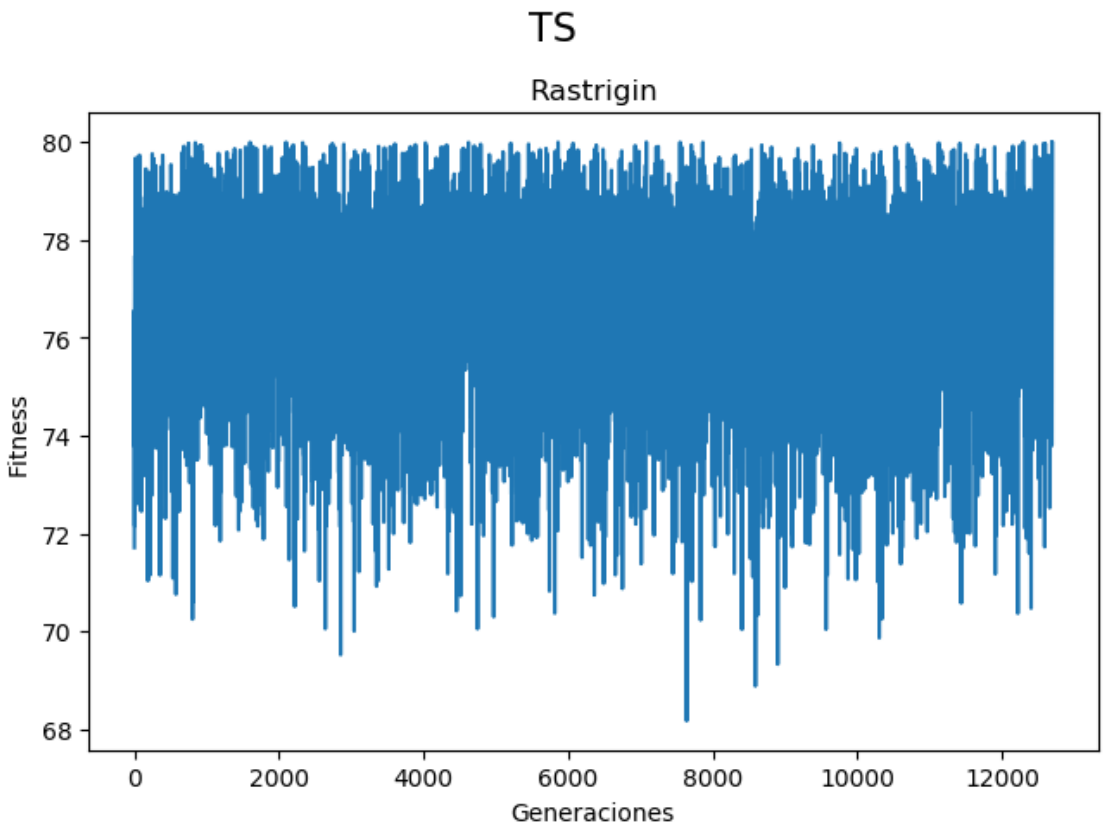
nament'
Global minimum found: [[-4.80312258 -4.76460666] [-0.57590283 4.52858924]]
after: 50000 generations.
Run: 3
Selection method: TS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[0.00082032 -0.00011719]]
after: 6967 generations.
Run: 4
Selection method: TS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[-0.00183595 0.00050782]]
after: 12697 generations.


TS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
500
1000
1500
2000
2500
3000
3500
4000


TS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
0
2000
4000
6000
8000
10000
12000


TS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
66
0
10000
20000
30000
40000
50000

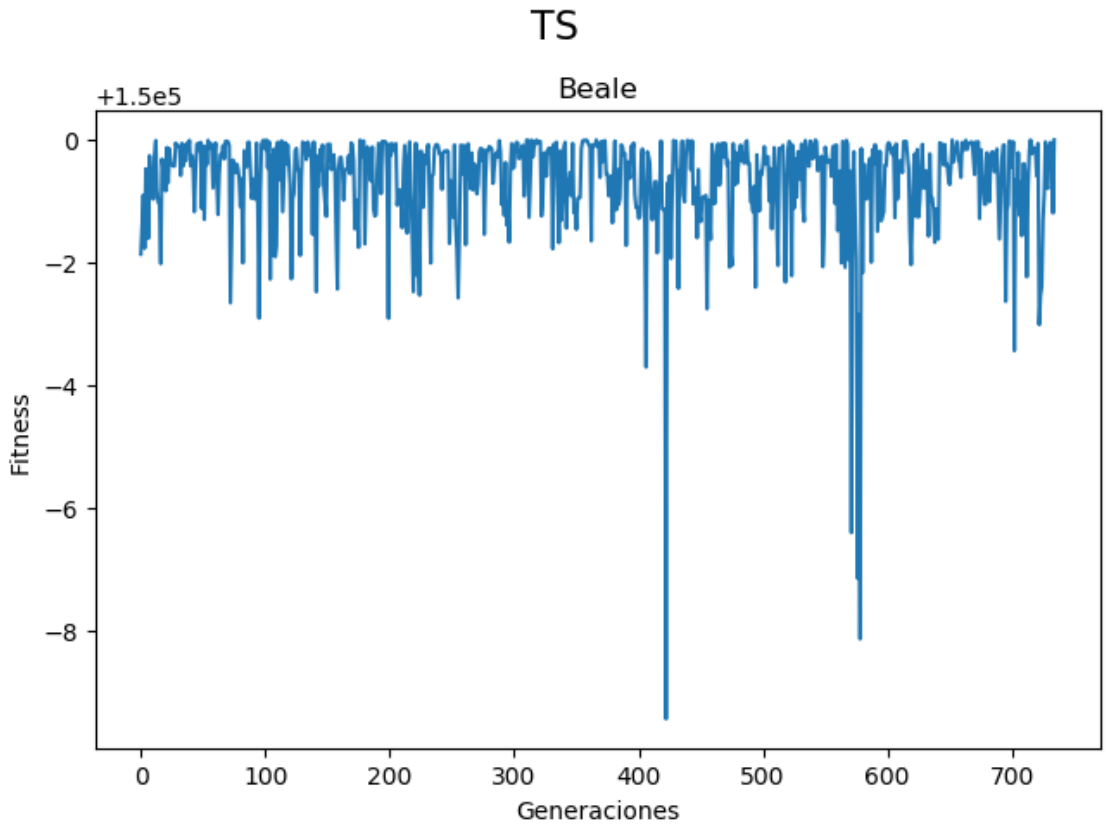

TS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
0
1000
2000
3000
4000
5000
6000
7000

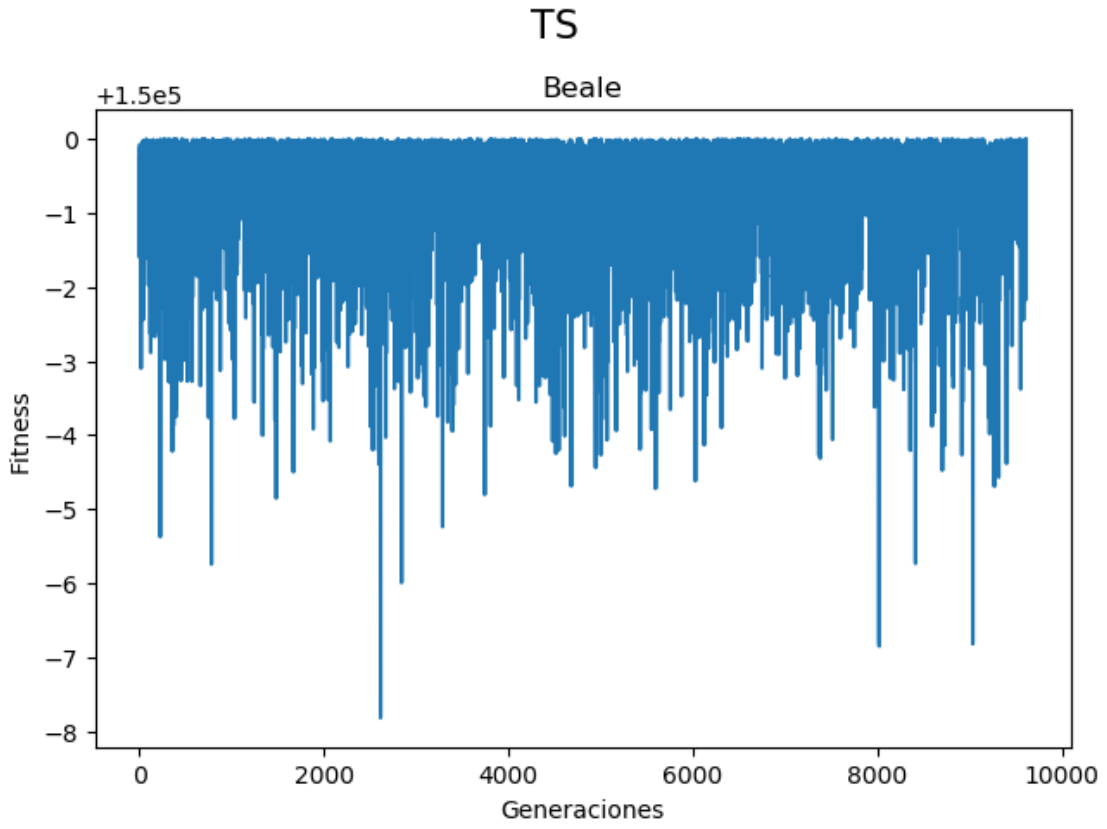


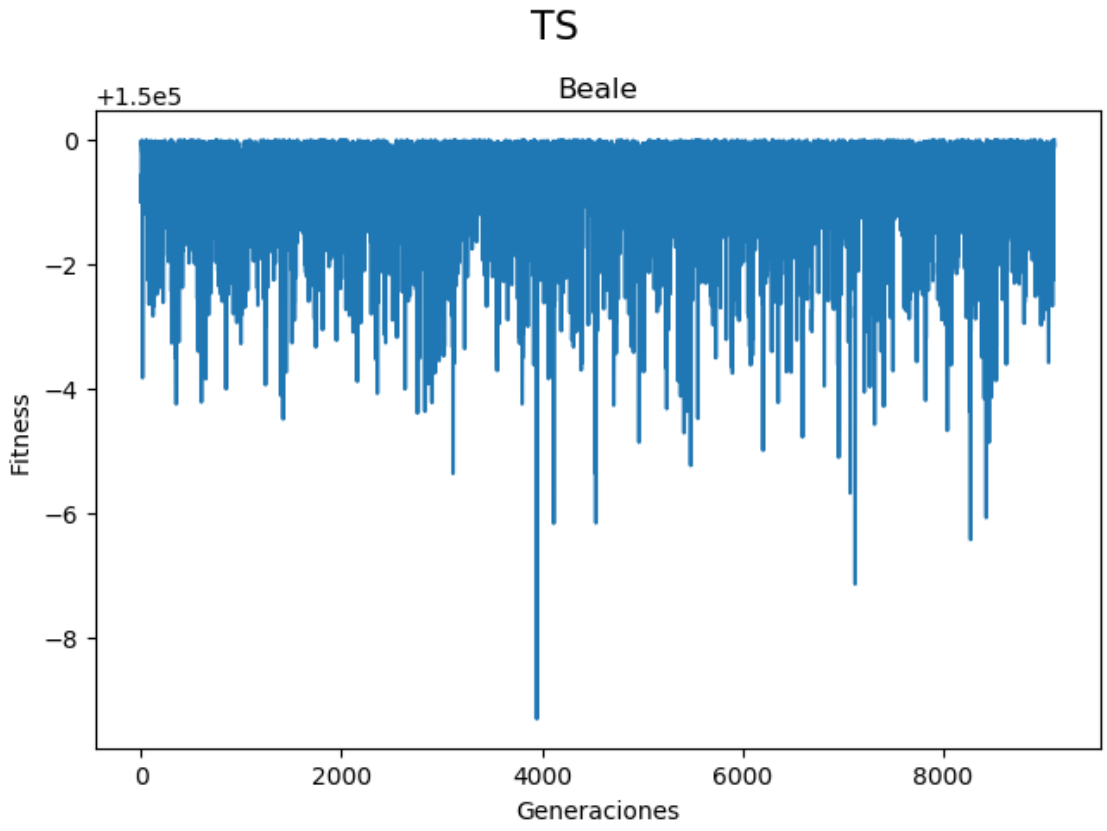
2.4.2. Beale

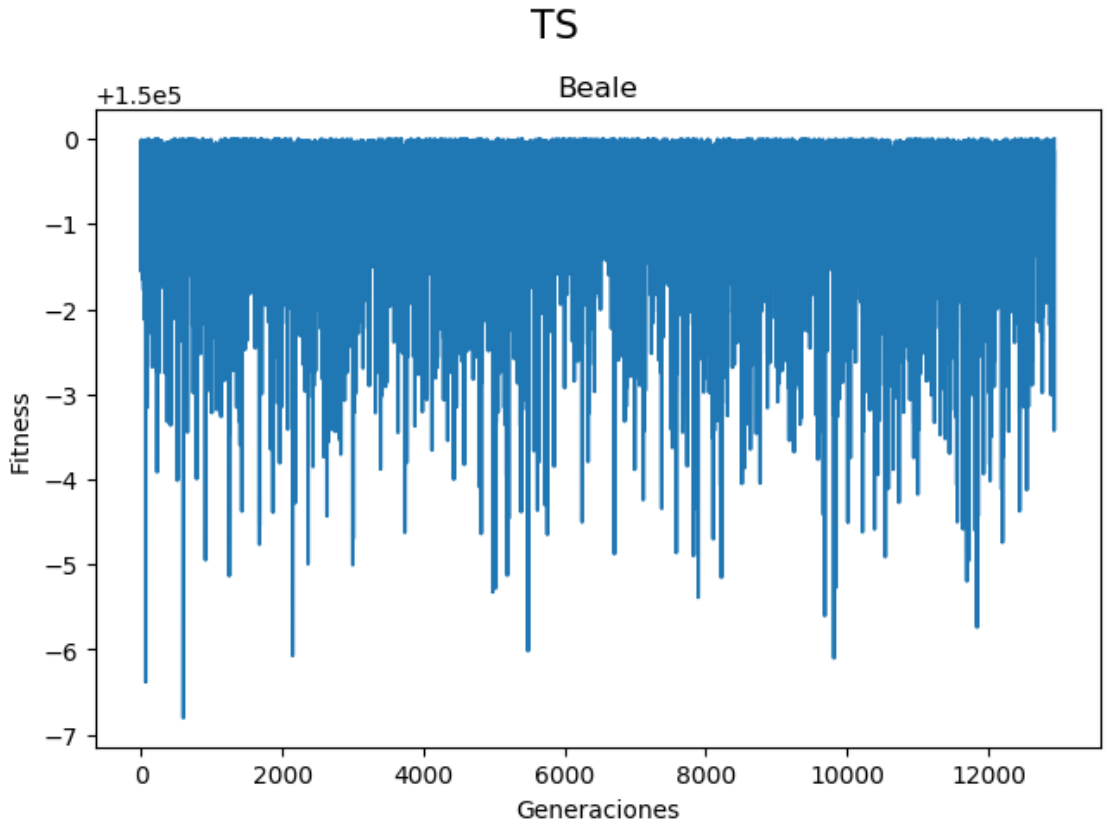
Run: 0
Selection method: TS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[3.00392535 0.50005341]]
after: 735 generations.
Run: 1
Selection method: TS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[2.99424358 0.4975128]]
after: 9610 generations.
Run: 2
Selection method: TS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[2.98813239 0.4961395]]

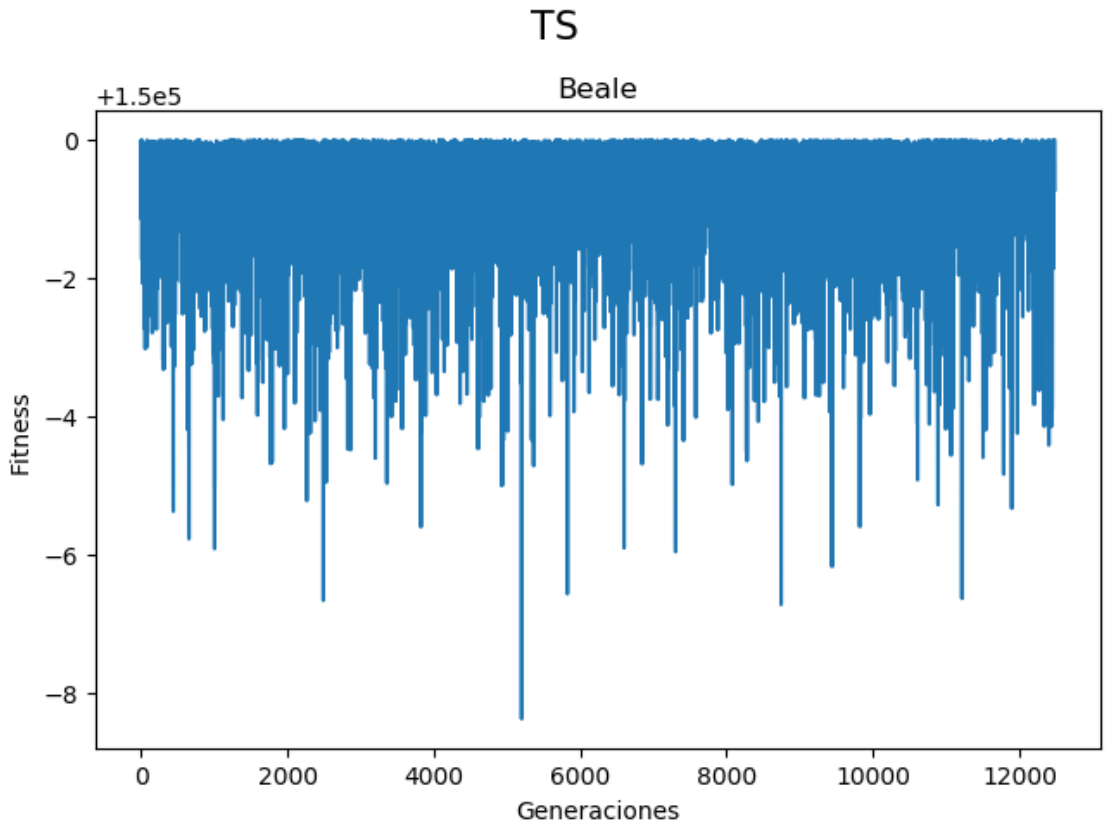
after: 9109 generations.
Run: 3
Selection method: TS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[2.98133454 0.49442287]]
after: 12934 generations.
Run: 4
Selection method: TS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[3.00358203 0.49977875]]
after: 12479 generations.


TS
Beale
+1.5e5
0
−2
−4
−6
−8
Fitness
0
100
200
300
400
500
600
700
Generaciones


TS
Beale
+1.5e5
0
−1
−2
−3
−4
−5
−6
−7
−8
Fitness
0
2000
4000
6000
8000
10000
Generaciones


TS
Beale
+1.5e5
0
−2
−4
−6
−8
Fitness
0
2000
4000
6000
8000
Generaciones

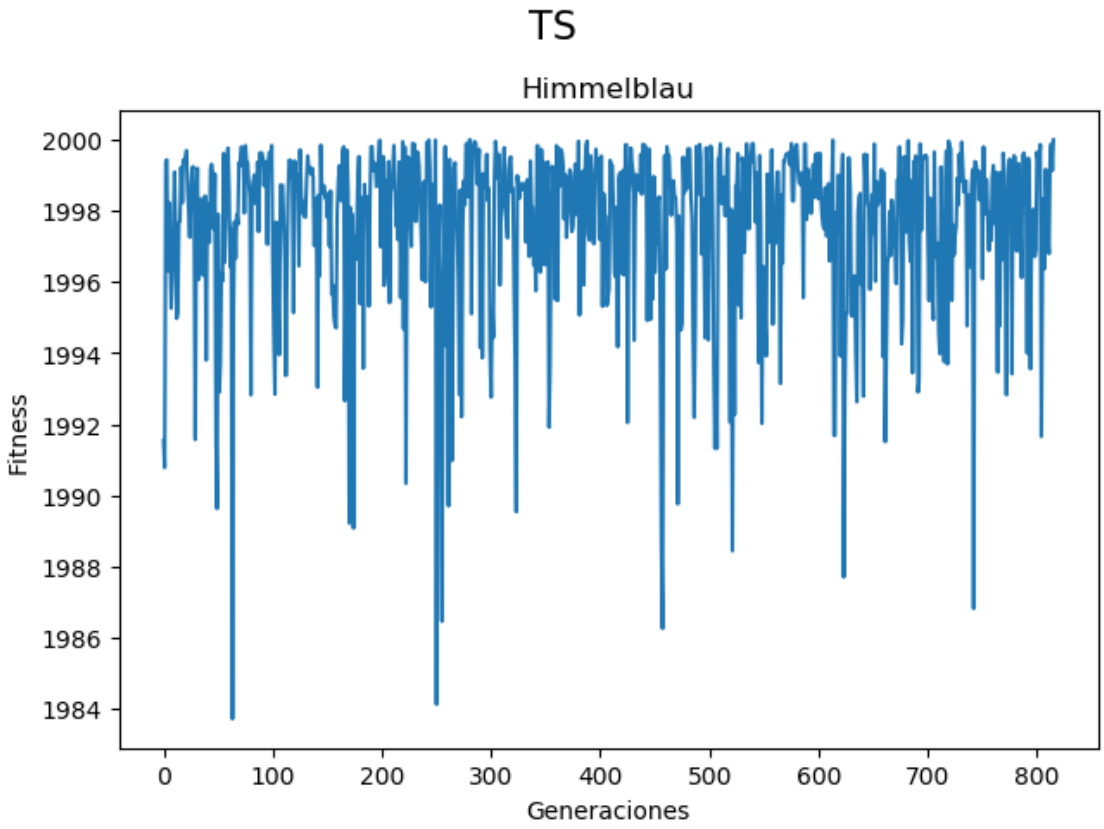

TS
Beale
+1.5e5
0
−1
−2
−3
−4
−5
−6
−7
Fitness
0
2000
4000
6000
8000
10000
12000
Generaciones

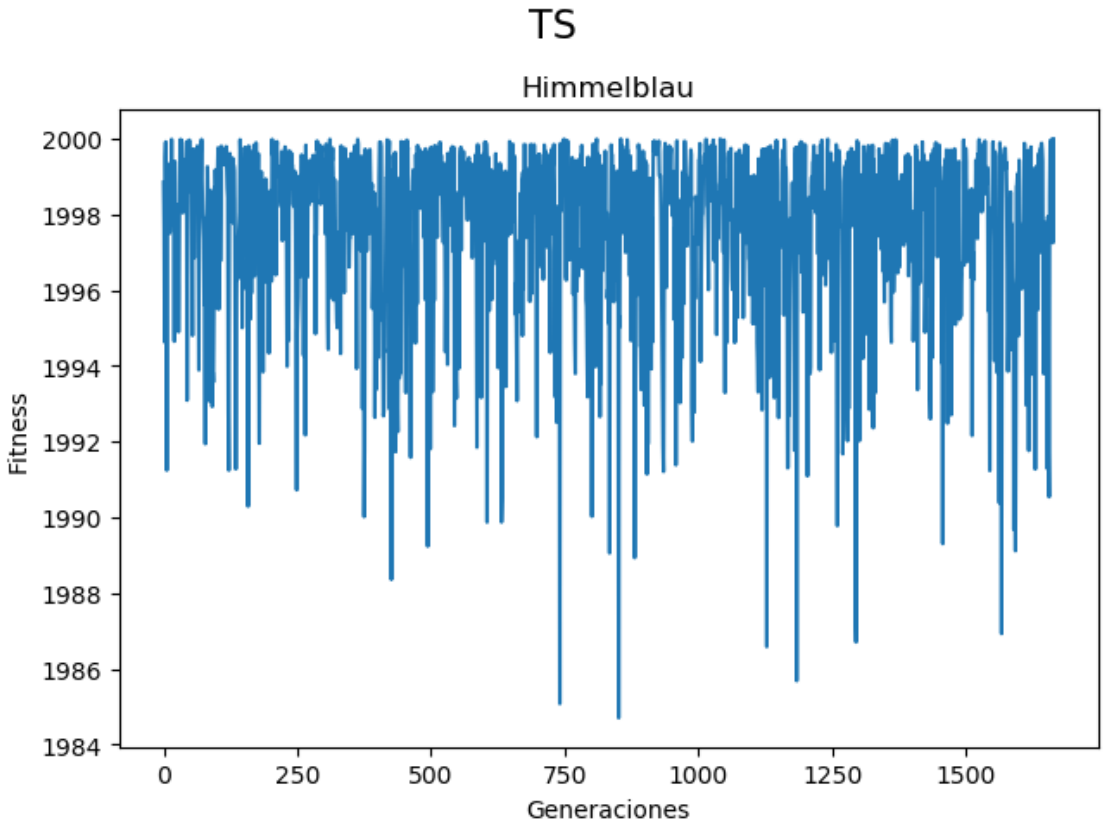


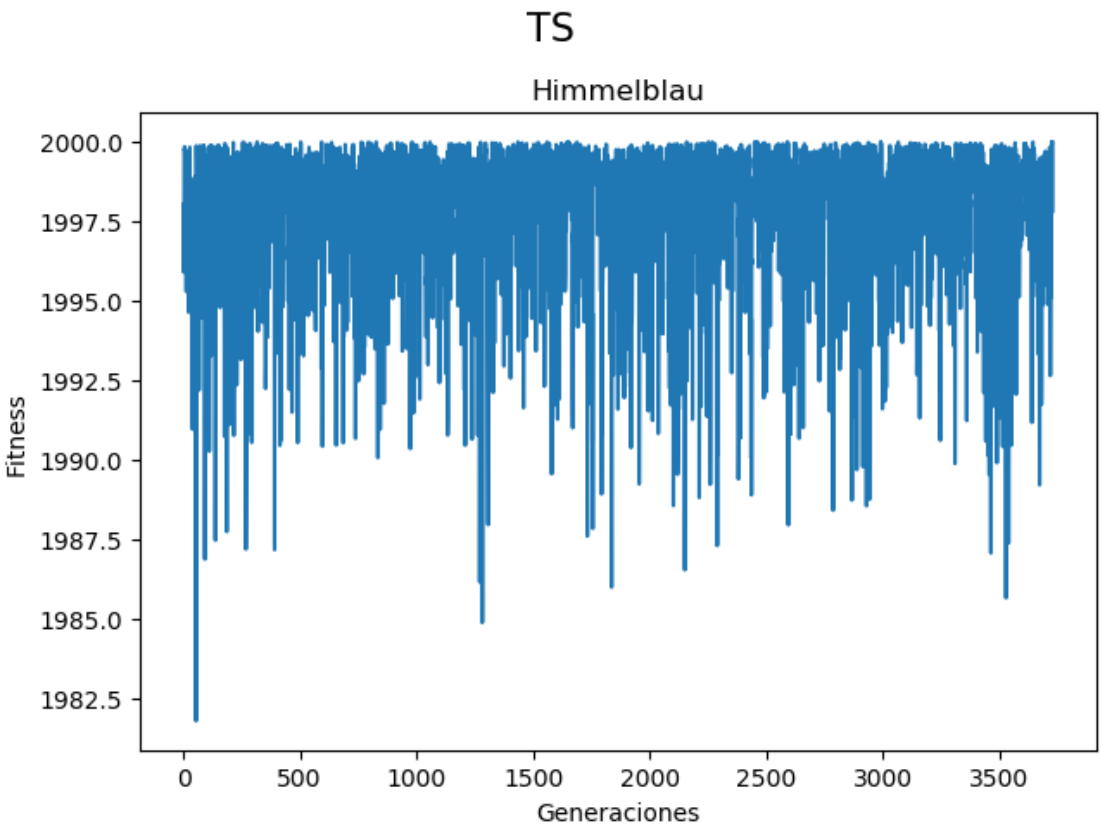
2.4.3. Himmelblau

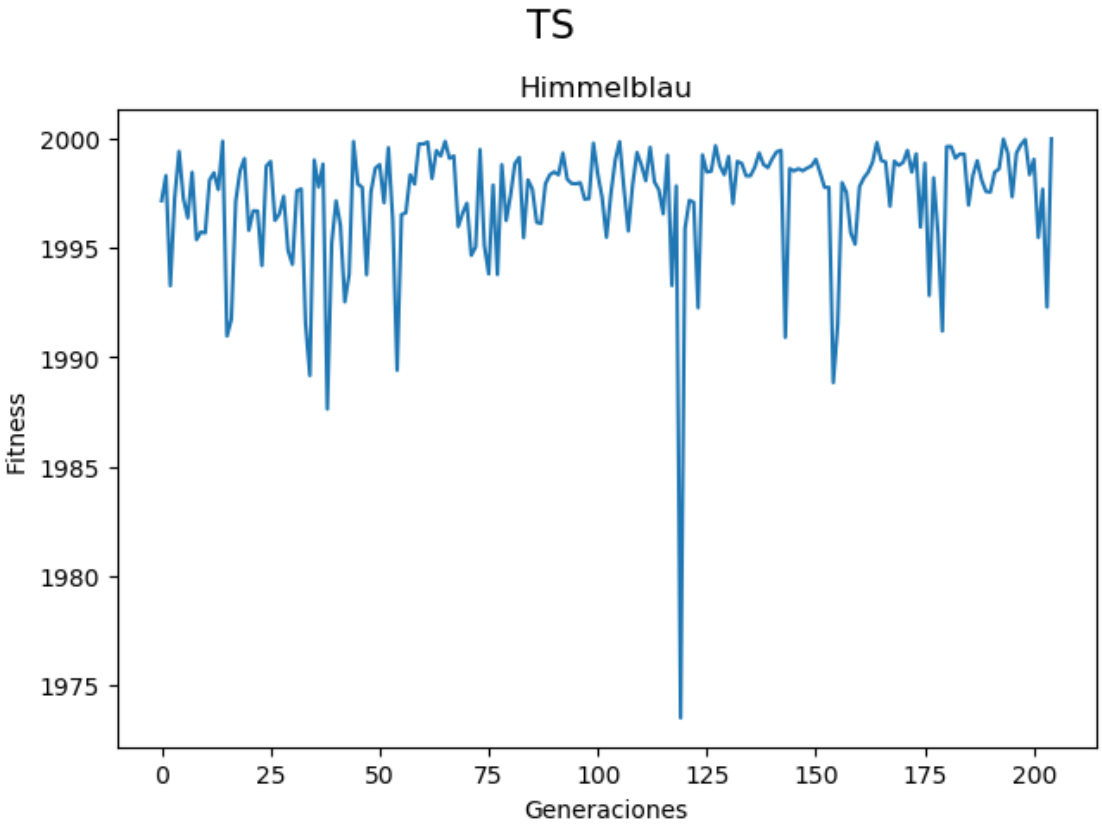
Run: 0
Selection method: TS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[3.00161744 1.99376674]]
after: 816 generations.
Run: 1
Selection method: TS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[3.00016785 2.00574498]]
after: 1665 generations.
Run: 2
Selection method: TS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[-3.78027939 -3.28230501]]

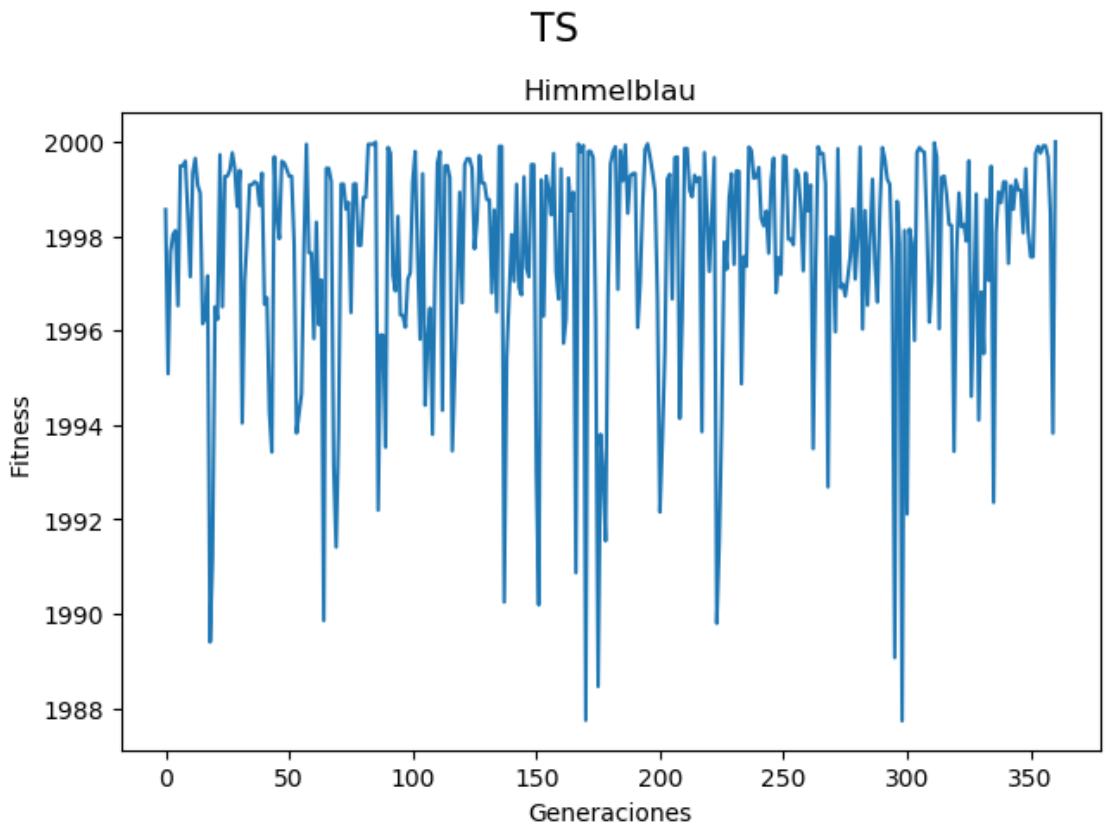
after: 3730 generations.
Run: 3
Selection method: TS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[2.99993896 1.99781035]]
after: 205 generations.
Run: 4
Selection method: TS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament'
Global minimum found: [[-2.80485386 3.13475139]]
after: 361 generations.


TS
Himmelblau
Fitness
2000
1998
1996
1994
1992
1990
1988
1986
1984
0
100
200
300
400
500
600
700
800
Generaciones


TS
Himmelblau
Fitness
2000
1998
1996
1994
1992
1990
1988
1986
1984
0
250
500
750
1000
1250
1500
Generaciones


TS
Himmelblau
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
Fitness
0
500
1000
1500
2000
2500
3000
3500
Generaciones

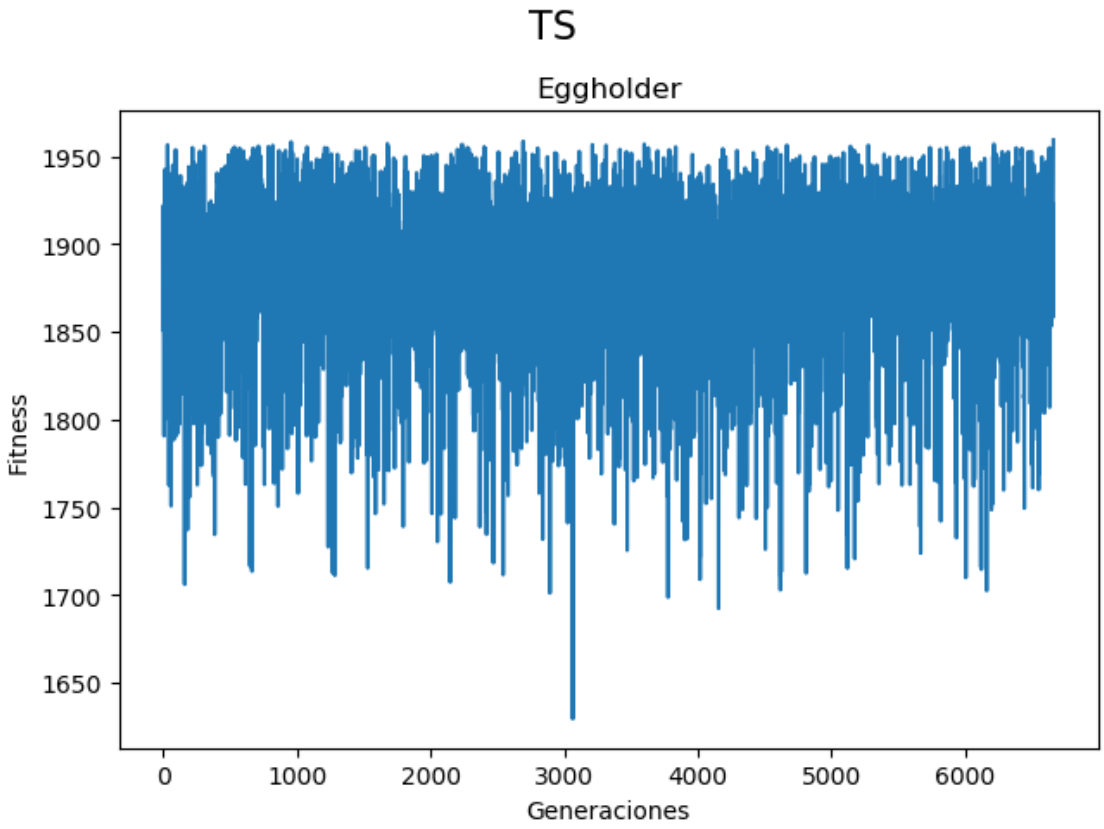

TS
Himmelblau
2000
1995
1990
1985
1980
1975
Fitness
0
25
50
75
100
125
150
175
200
Generaciones

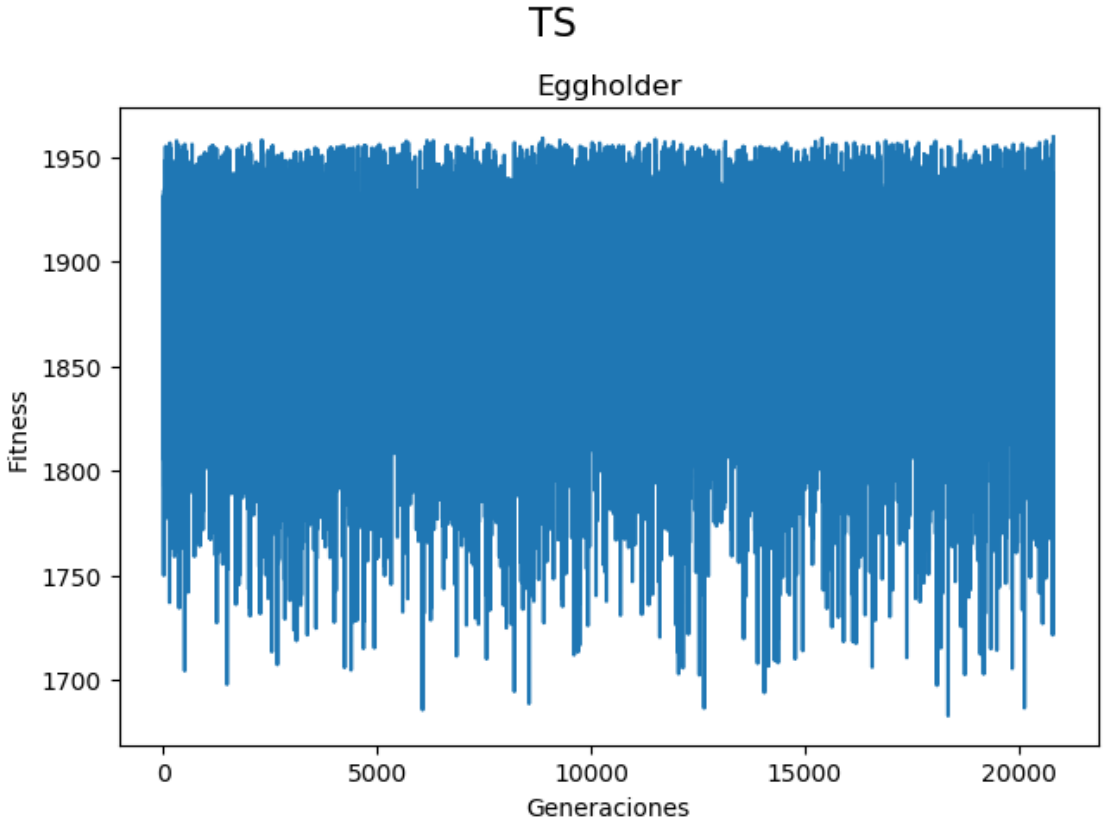


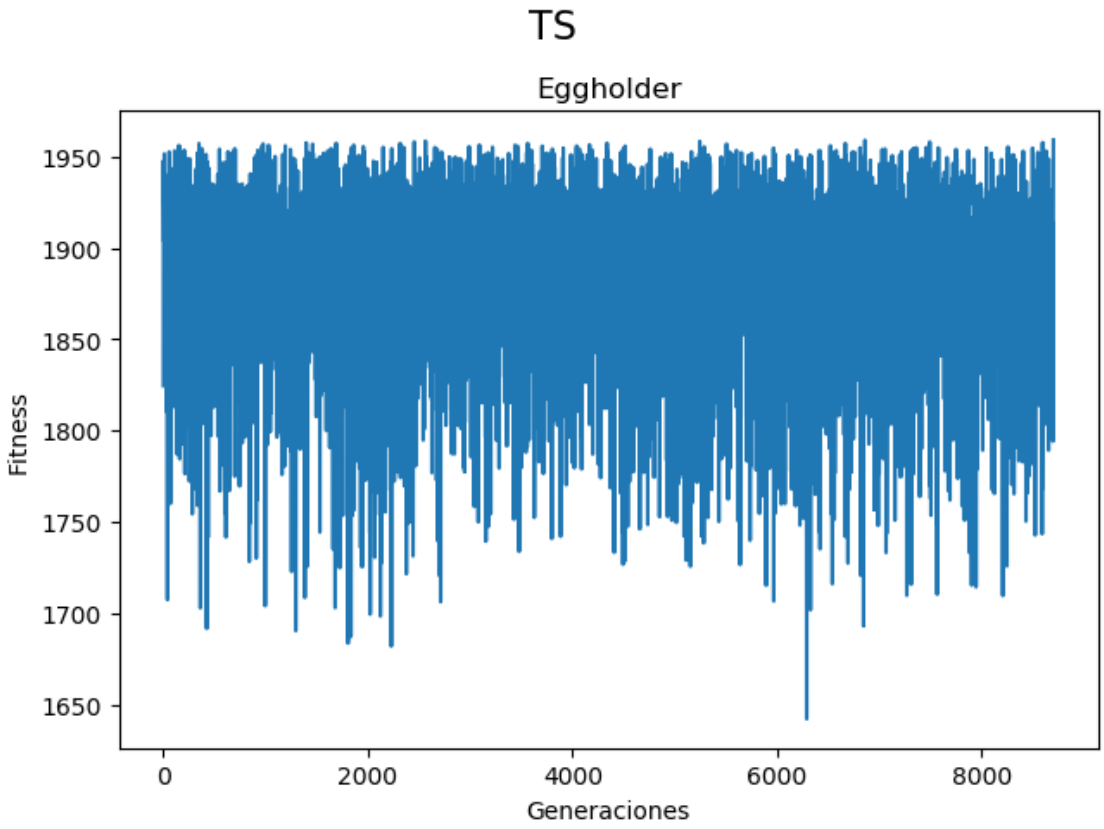
2.4.4. Eggholder

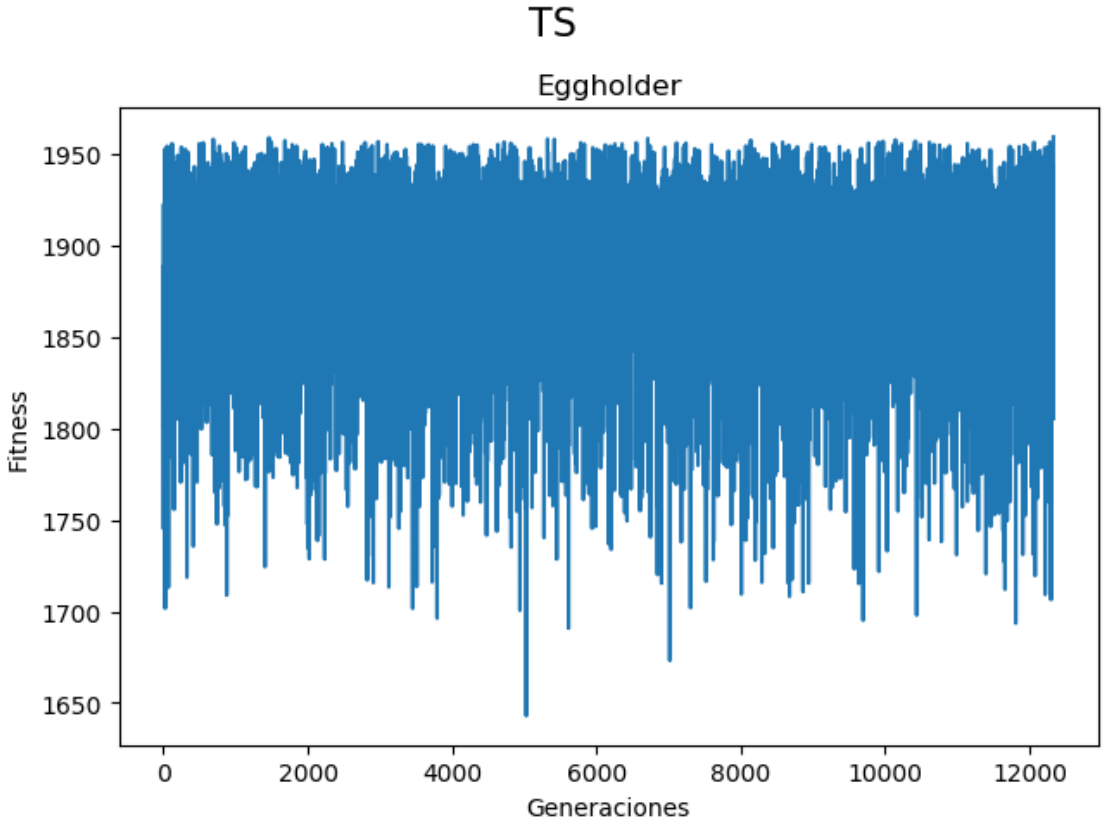
Run: 0
Selection method: TS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament'
Global minimum found: [[511.99609375 404.54491547]]
after: 6660 generations.
Run: 1
Selection method: TS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament'
Global minimum found: [[511.9956665 403.92546963]]
after: 20823 generations.
Run: 2
Selection method: TS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament'
Global minimum found: [[511.87811279 403.71941493]]

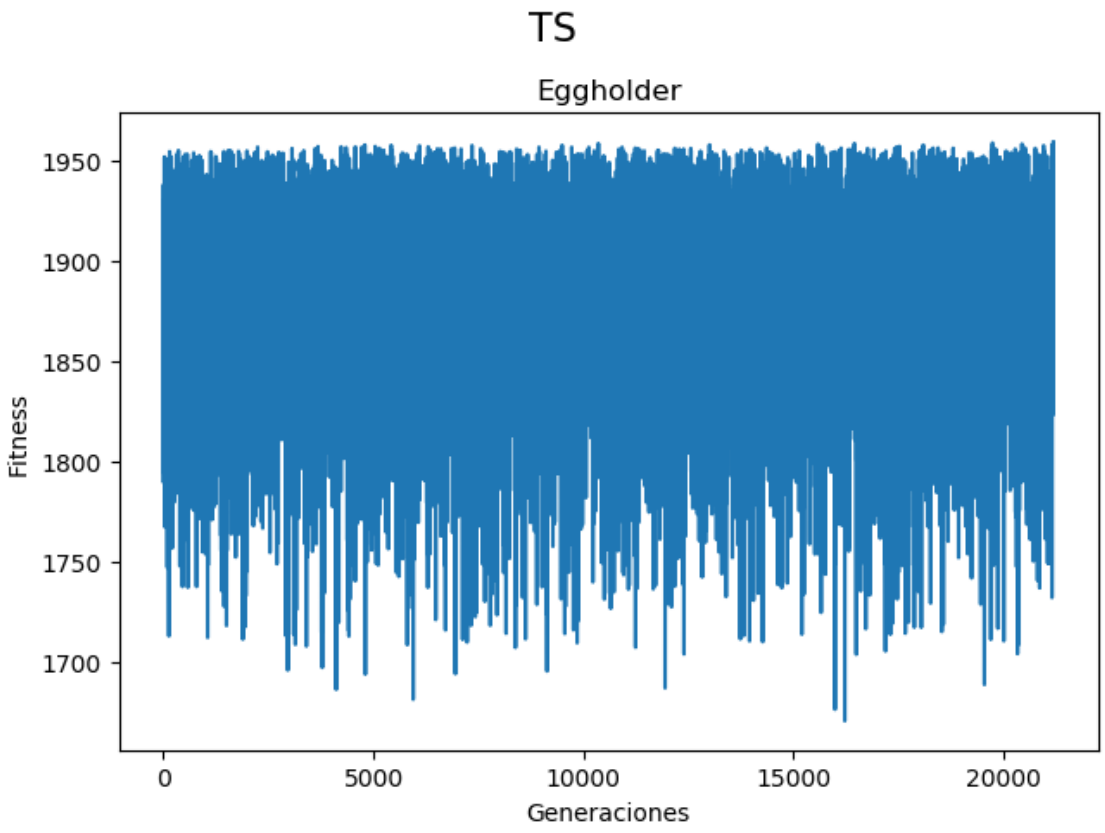
after: 8703 generations.
Run: 3
Selection method: TS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament'
Global minimum found: [[511.99249268 403.72503017]]
after: 12332 generations.
Run: 4
Selection method: TS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament'
Global minimum found: [[511.91851806 404.09148527]]
after: 21198 generations.


TS
Eggholder
Fitness
1950
1900
1850
1800
1750
1700
1650
0
1000
2000
3000
4000
5000
6000
Generaciones


TS
Eggholder
Fitness
1950
1900
1850
1800
1750
1700
0
5000
10000
15000
20000
Generaciones


TS
Eggholder
Fitness
1950
1900
1850
1800
1750
1700
1650
0
2000
4000
6000
8000
Generaciones


TS
Eggholder
Fitness
1950
1900
1850
1800
1750
1700
1650
0
2000
4000
6000
8000
10000
12000
Generaciones



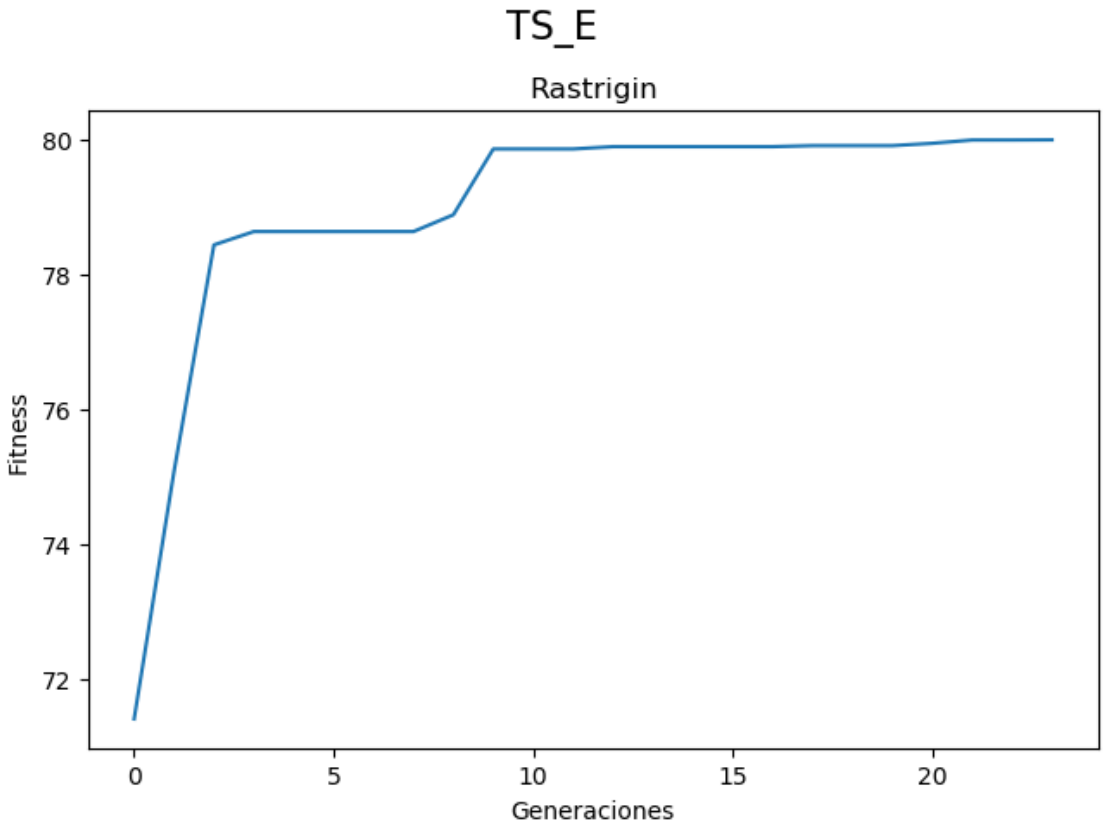
2.5. Selección por torneo, con elitismo

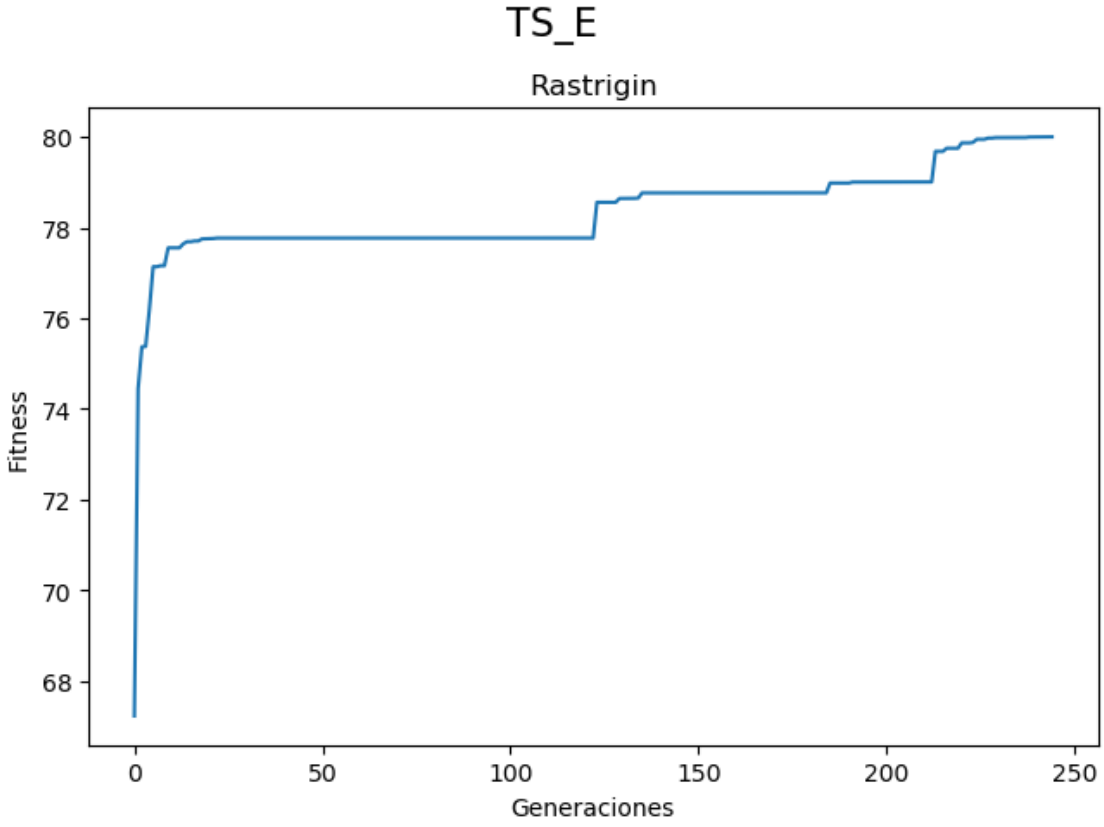
2.5.1. Rastrigin

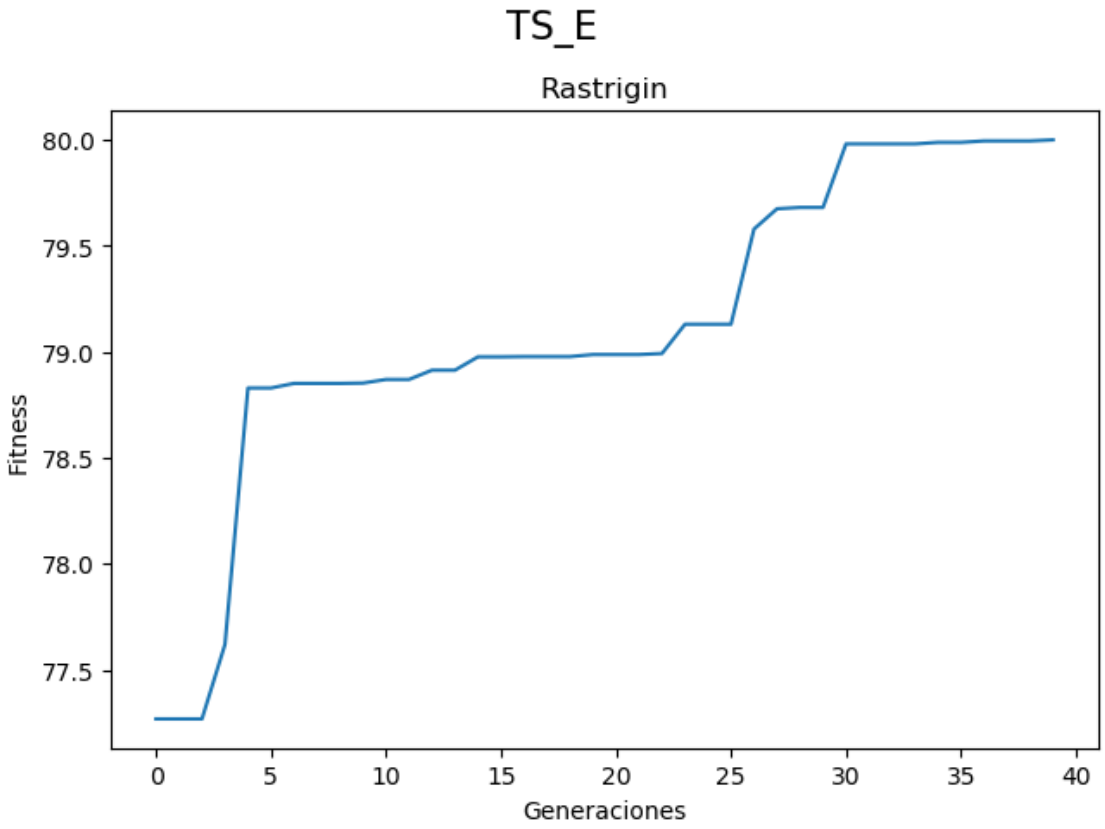
Run: 0
Selection method: TS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-0.00058594 0.00121095]]
after: 24 generations.
Run: 1
Selection method: TS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-0.00207033 0.00058594]]
after: 245 generations.
Run: 2
Selection method: TS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tour-

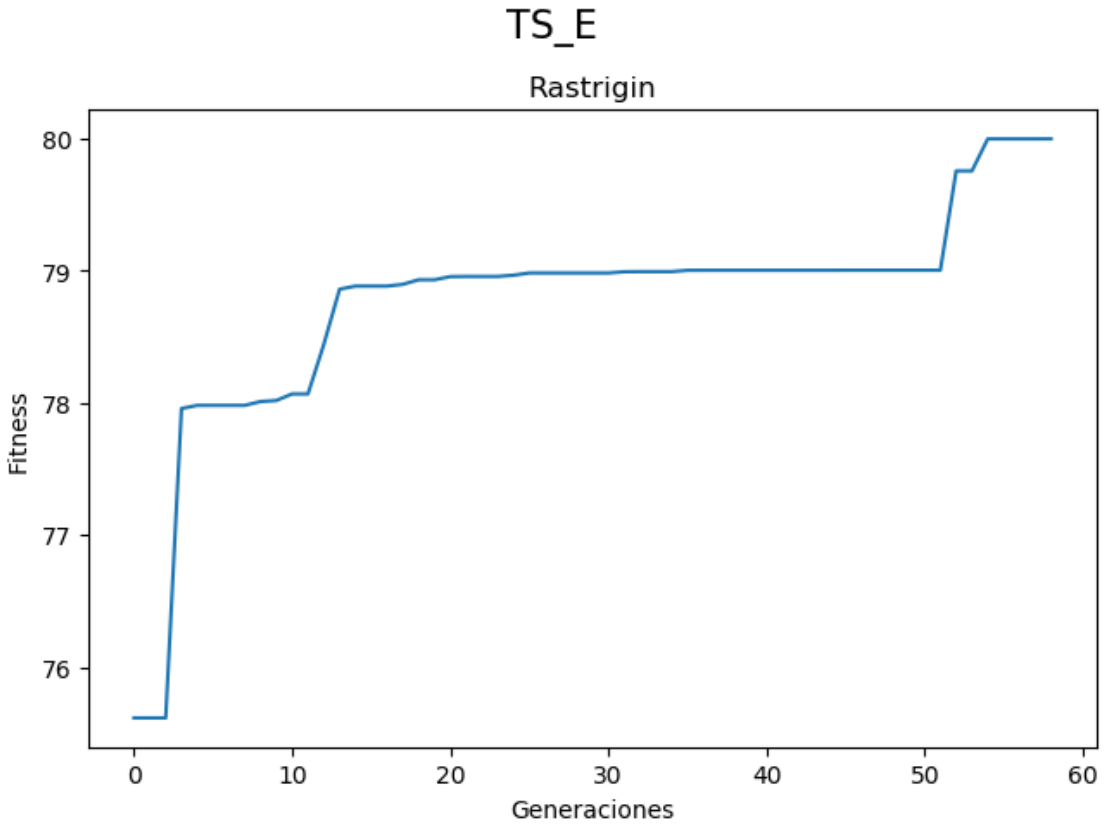
nament', 'elitism': 0.1
Global minimum found: [[0.00019531 -0.00128907]]
after: 40 generations.
Run: 3
Selection method: TS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-1.75782591e-03 -1.28907233e-03] [-2.07032830e-03 -3.90627980e-05]]
after: 59 generations.
Run: 4
Selection method: TS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[0.00011719 -0.00082032]]
after: 41 generations.
Probabilidad de mutación: 0.0625
Elitismo: 0.2
Número de iteraciones: 19
Mínimo global:

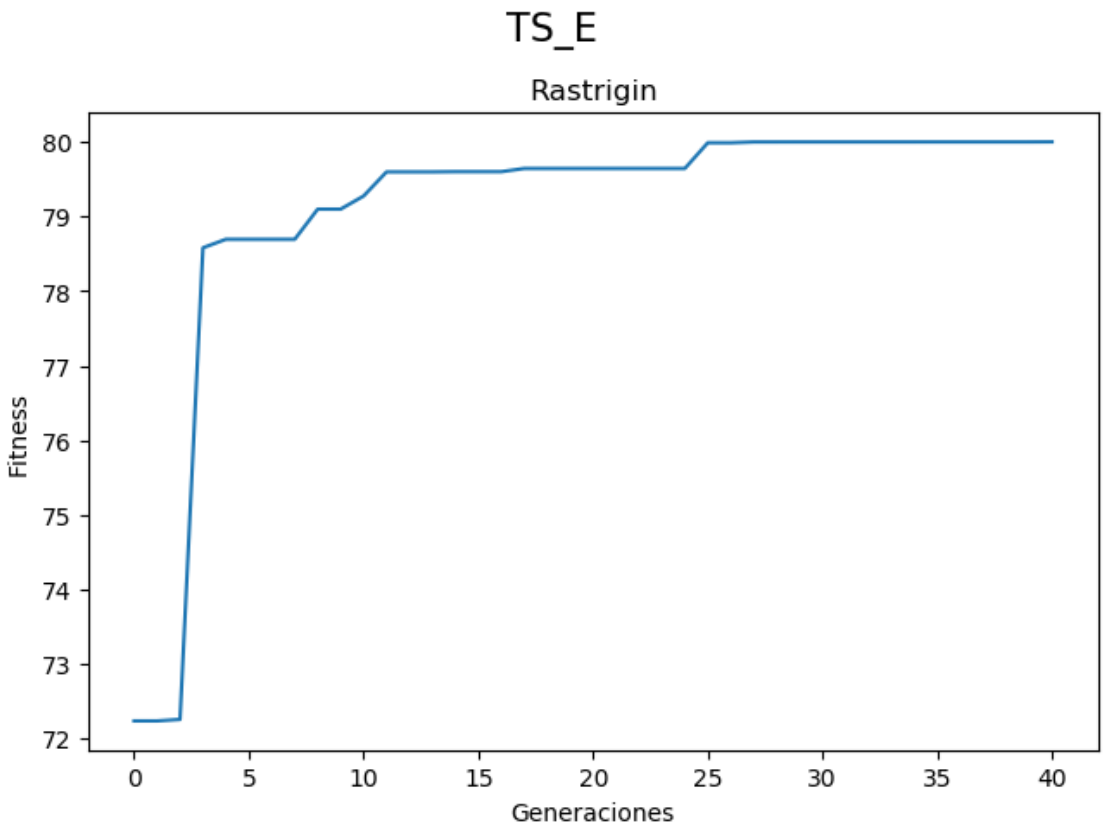
[-0.00097657 -0.00082032
]


TS_E
Rastrigin
Fitness
Generaciones
80
78
76
74
72
0
5
10
15
20


TS_E
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
0
50
100
150
200
250


TS_E
Rastrigin
Fitness
Generaciones
80.0
79.5
79.0
78.5
78.0
77.5
0
5
10
15
20
25
30
35
40

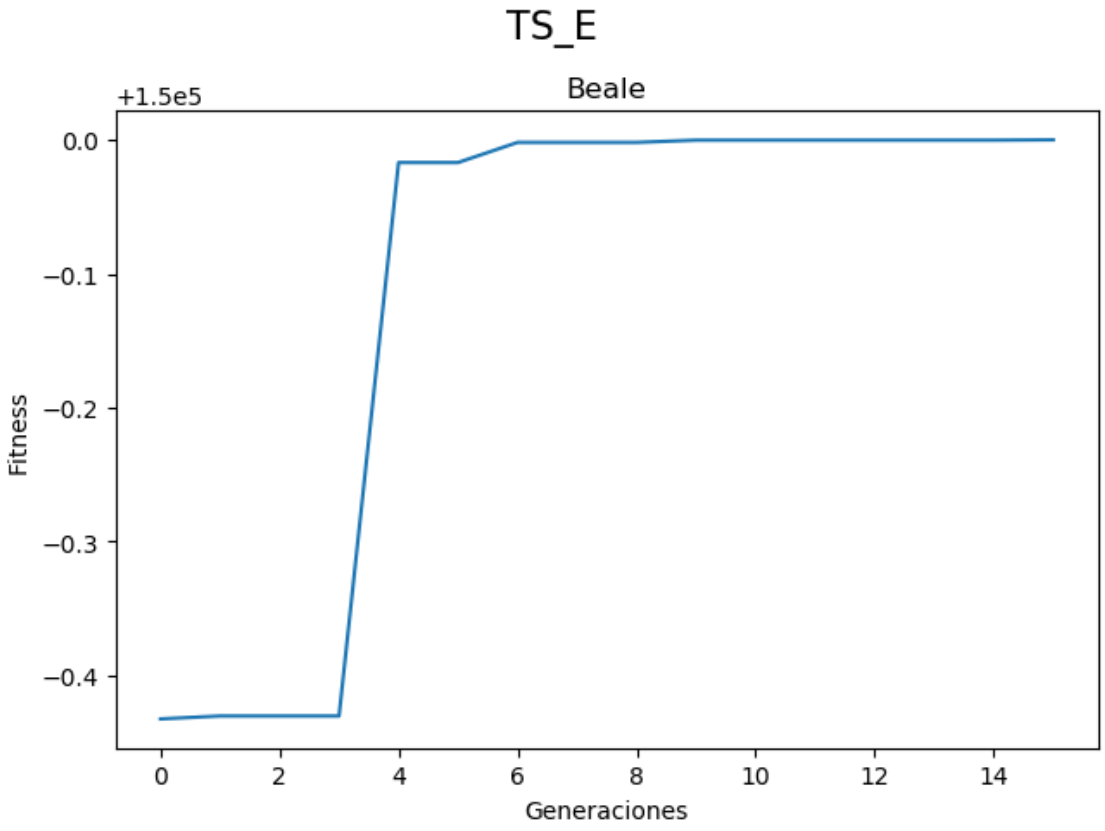

TS_E
Rastrigin
Fitness
Generaciones
80
79
78
77
76
0
10
20
30
40
50
60

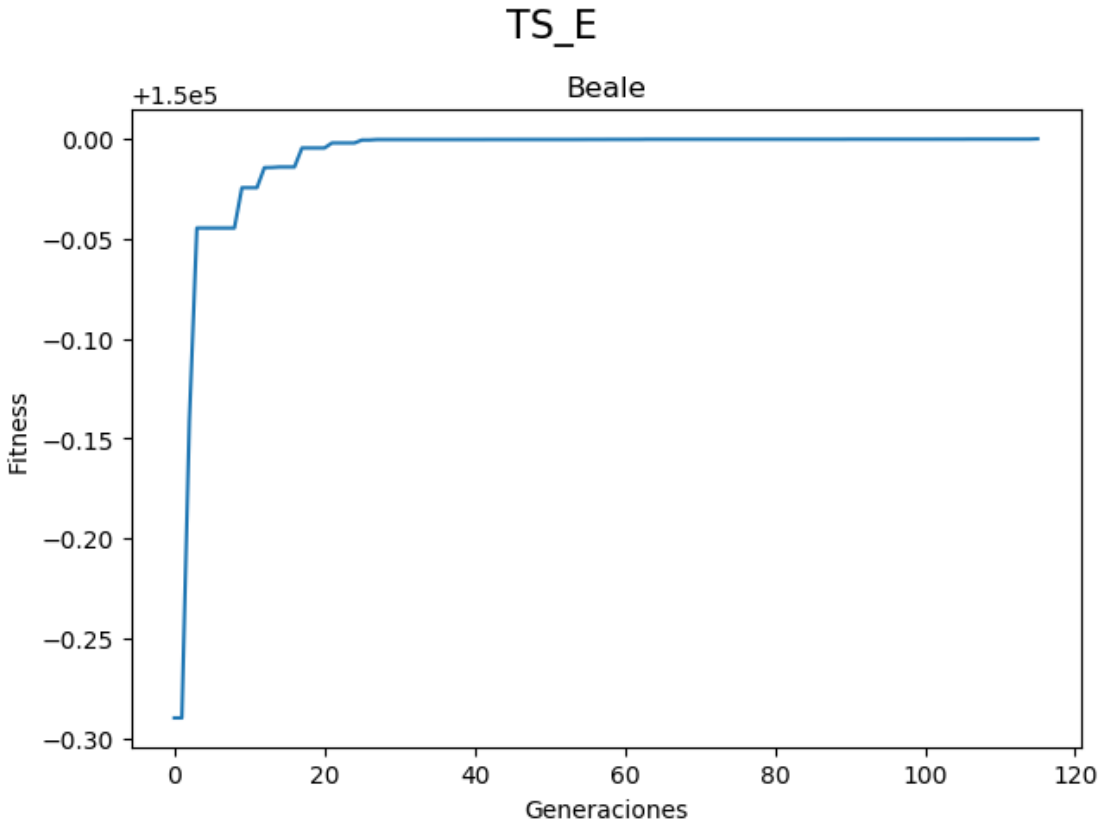


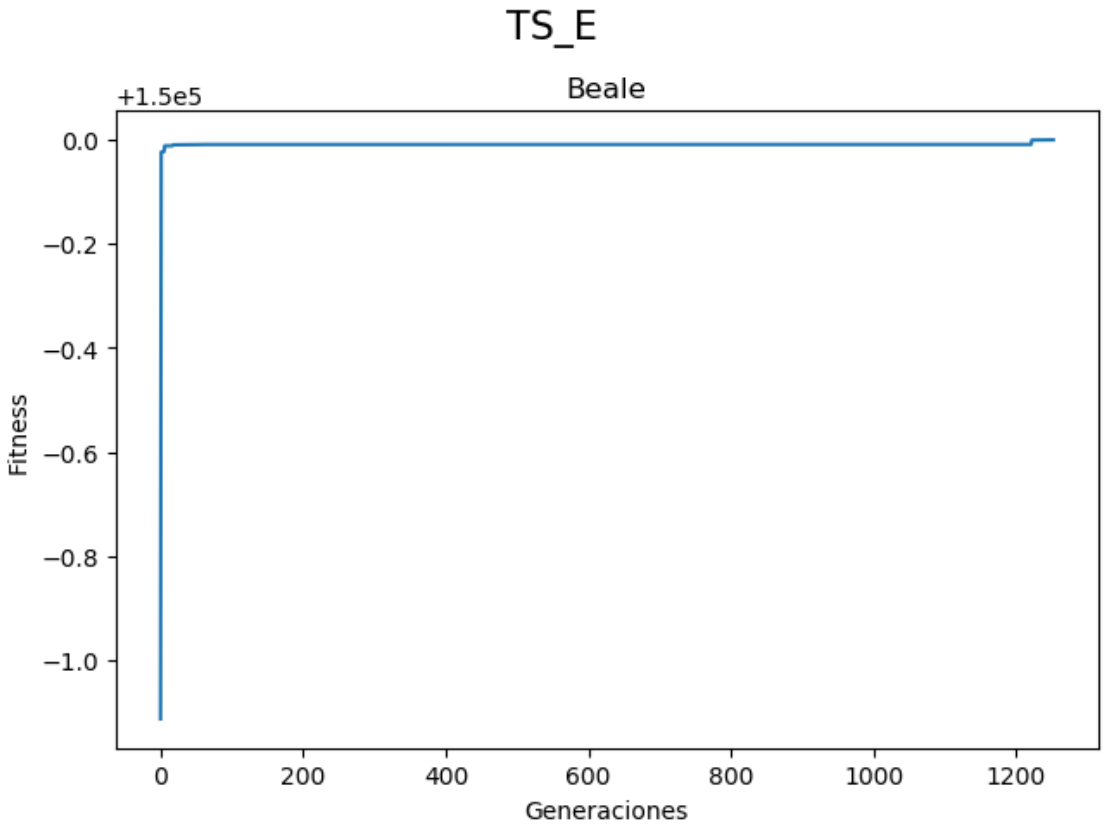
2.5.2. Beale

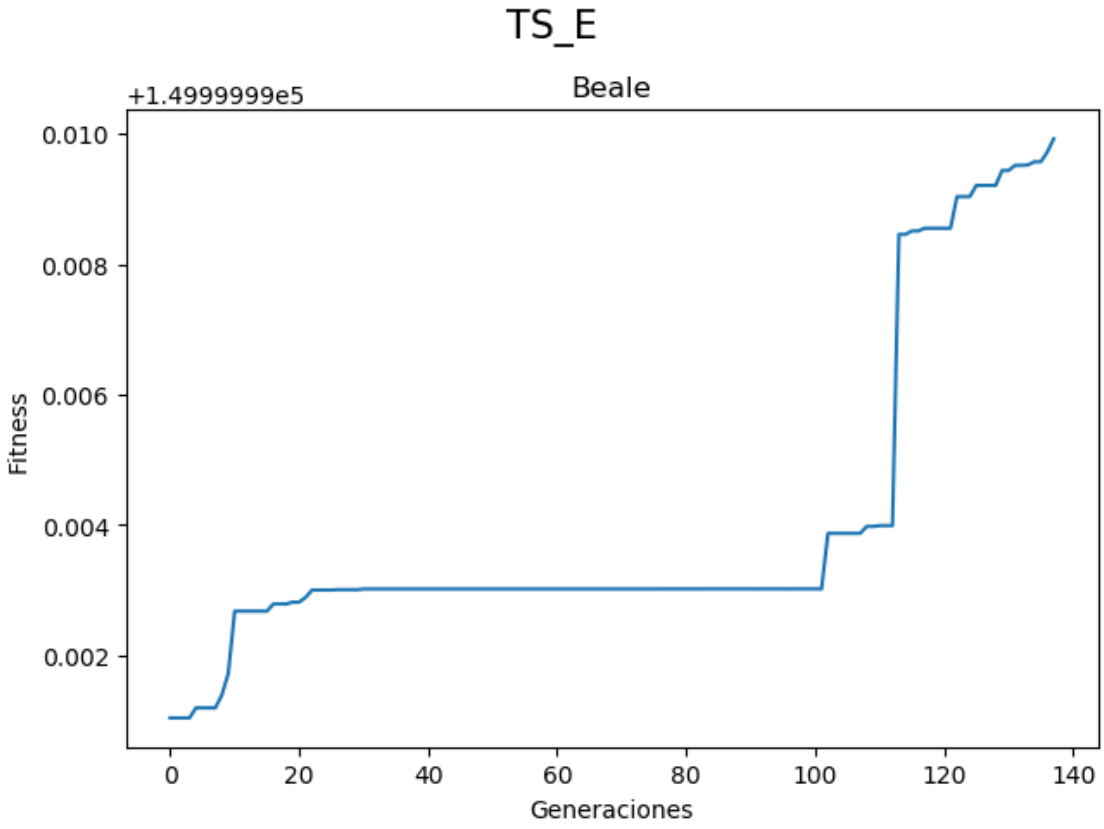
Run: 0
Selection method: TS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.2
Global minimum found: [[3.00625997 0.50225069]]
after: 16 generations.
Run: 1
Selection method: TS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.2
Global minimum found: [[3.00296404 0.50101472]]
after: 116 generations.
Run: 2
Selection method: TS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.2
Global minimum found: [[2.98229585 0.49668882]]

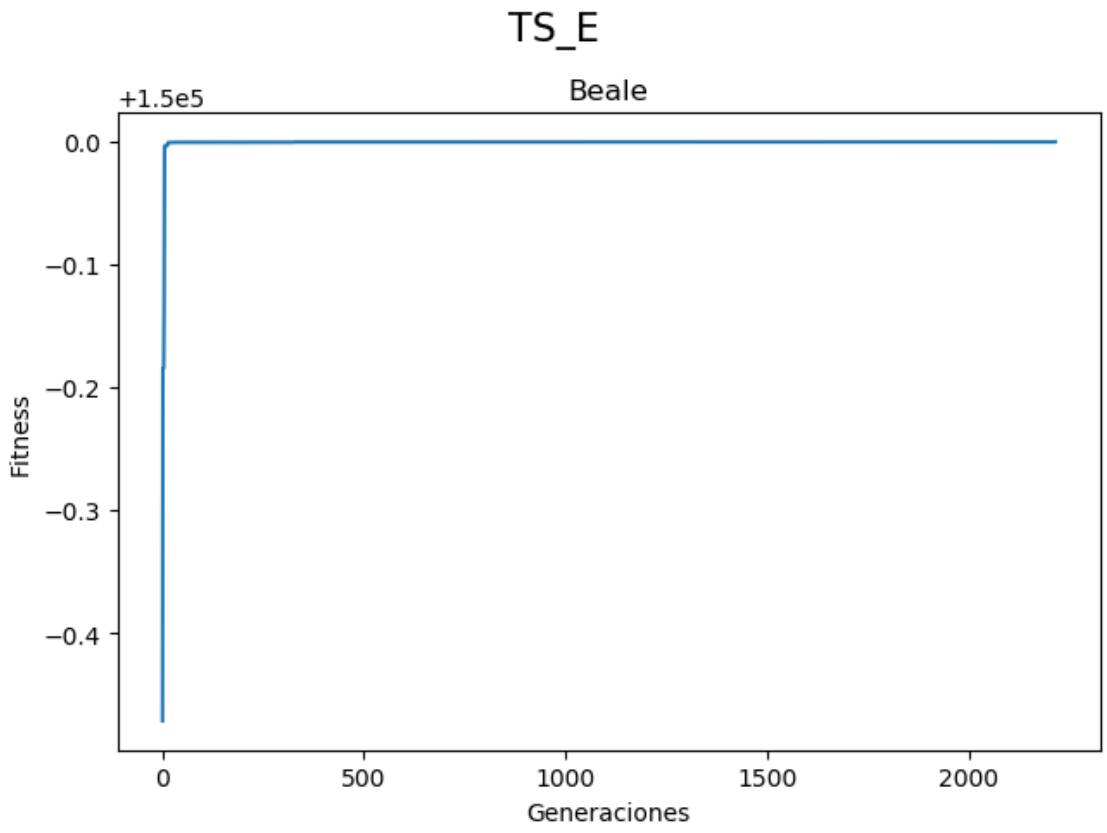
after: 1253 generations.
Run: 3
Selection method: TS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.2
Global minimum found: [[3.00420001 0.49922943]]
after: 138 generations.
Run: 4
Selection method: TS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.2
Global minimum found: [[2.98167787 0.4959335]]
after: 2216 generations.


TS_E
Beale
+1.5e5
0.0
−0.1
−0.2
−0.3
−0.4
Fitness
0
2
4
6
8
10
12
14
Generaciones


TS_E
Beale
+1.5e5
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
Fitness
0
20
40
60
80
100
120
Generaciones


TS_E
Beale
+1.5e5
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
Fitness
0
200
400
600
800
1000
1200
Generaciones

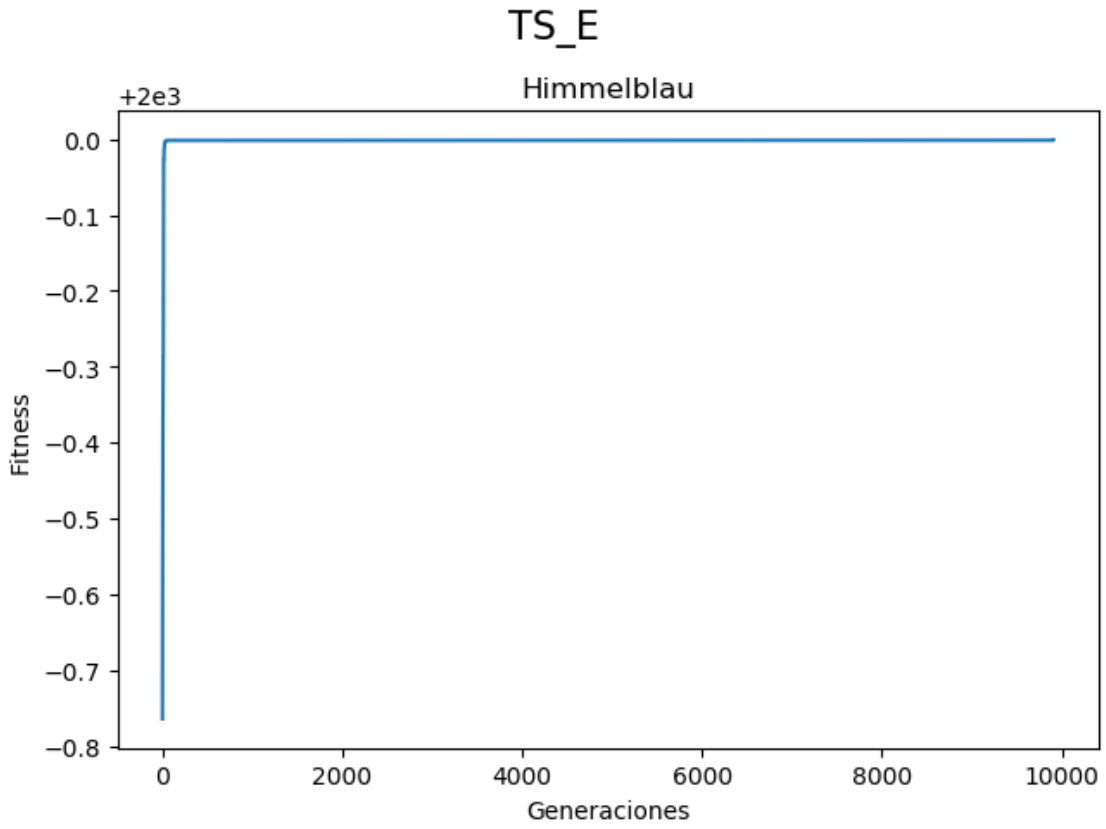

TS_E
Beale
+1.4999999e5
0.010
0.008
0.006
0.004
0.002
Fitness
0
20
40
60
80
100
120
140
Generaciones

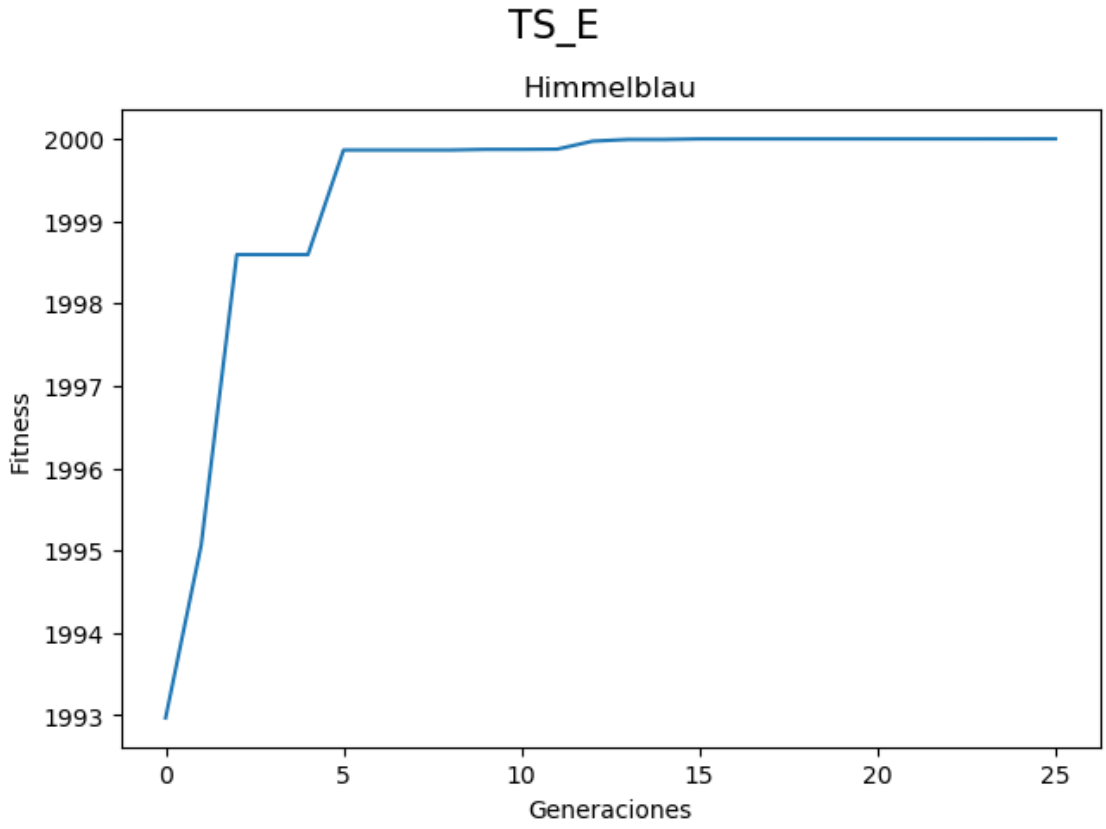


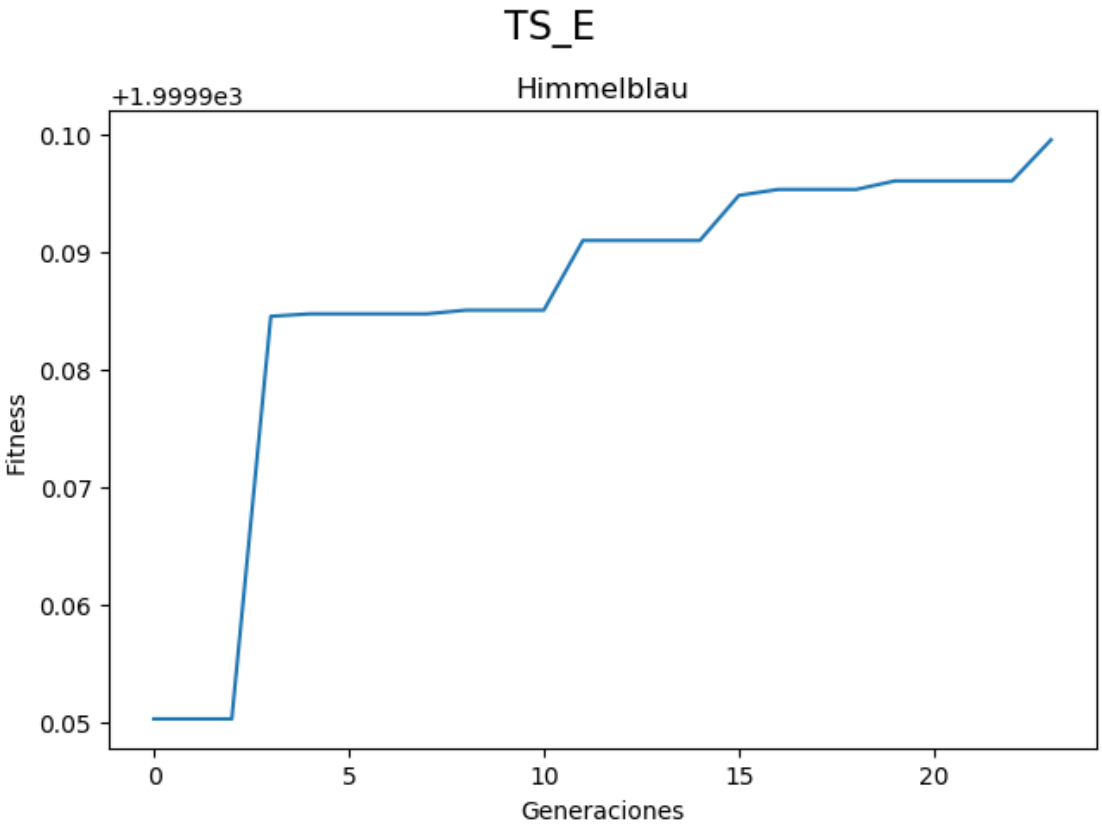
2.5.3. Himmelblau

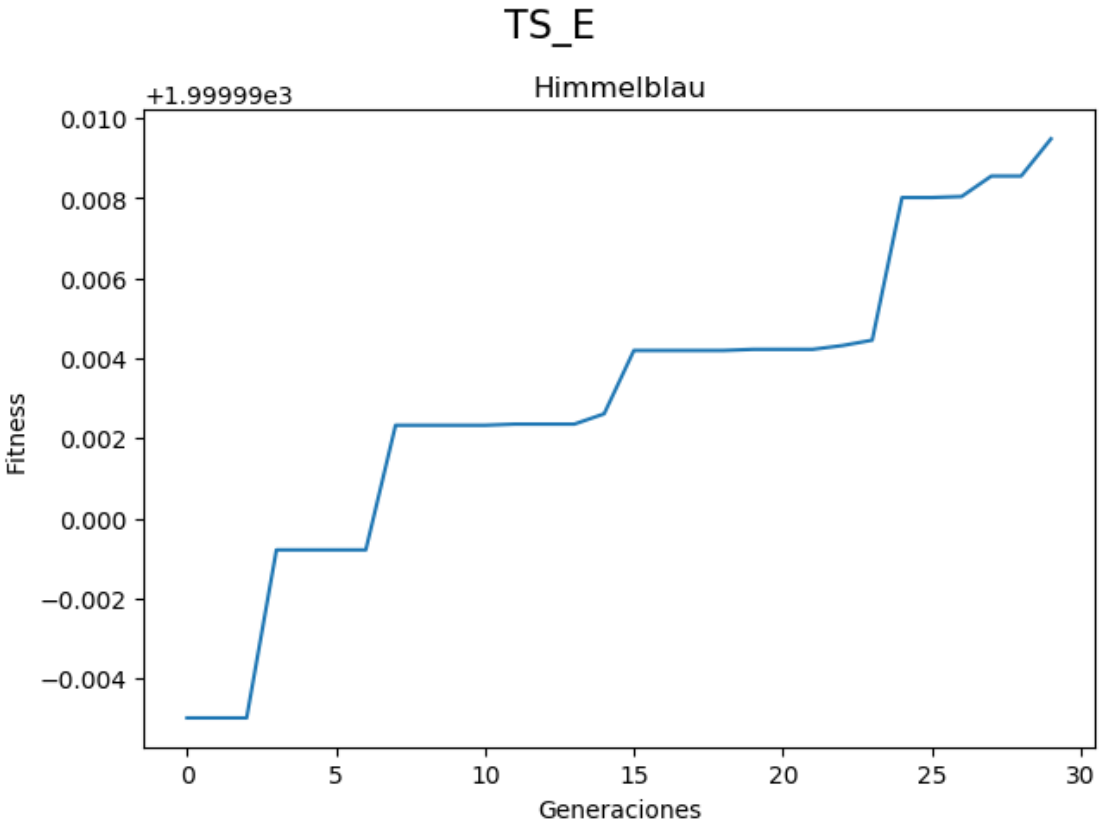
Run: 0
Selection method: TS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-2.80500645 3.13597211]]
after: 9909 generations.
Run: 1
Selection method: TS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-2.80096284 3.13025002]]
after: 26 generations.
Run: 2
Selection method: TS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-3.77776167 -3.28001617]]

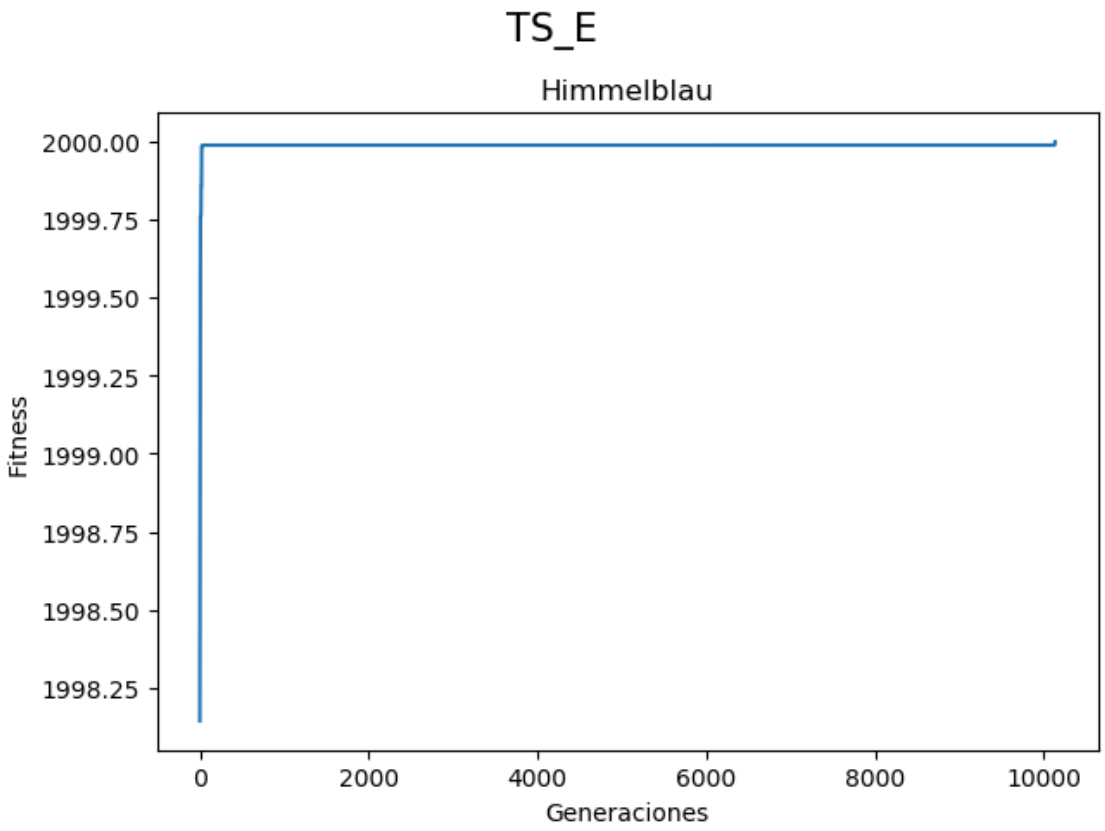
after: 24 generations.
Run: 3
Selection method: TS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[-2.80523533 3.13490398] [-2.80752417 3.13490398]]
after: 30 generations.
Run: 4
Selection method: TS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[3.58481281 -1.84575535]]
after: 10135 generations.


TS_E
Himmelblau
+2e3
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
Fitness
0
2000
4000
6000
8000
10000
Generaciones


TS_E
Himmelblau
2000
1999
1998
1997
1996
1995
1994
1993
Fitness
0
5
10
15
20
25
Generaciones


TS_E
Himmelblau
+1.9999e3
0.10
0.09
0.08
0.07
0.06
0.05
Fitness
0
5
10
15
20
Generaciones

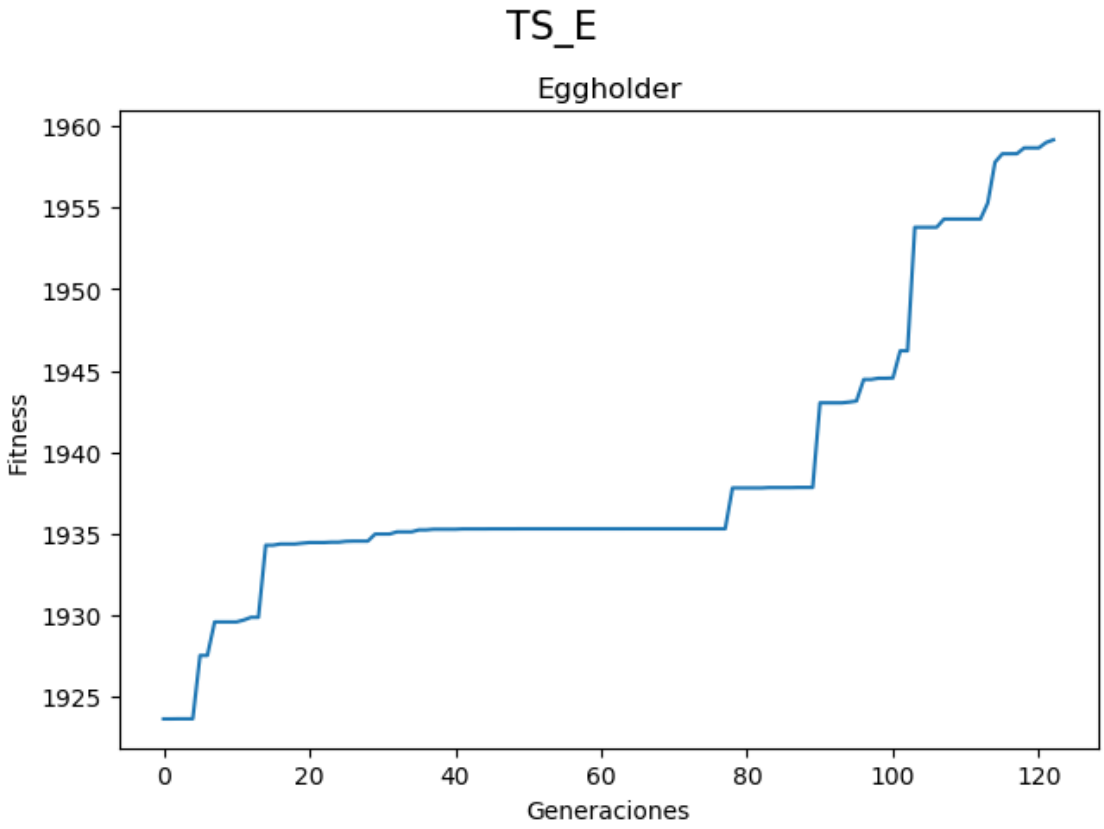

TS_E
Himmelblau
+1.99999e3
0.010
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
Fitness
0
5
10
15
20
25
30
Generaciones

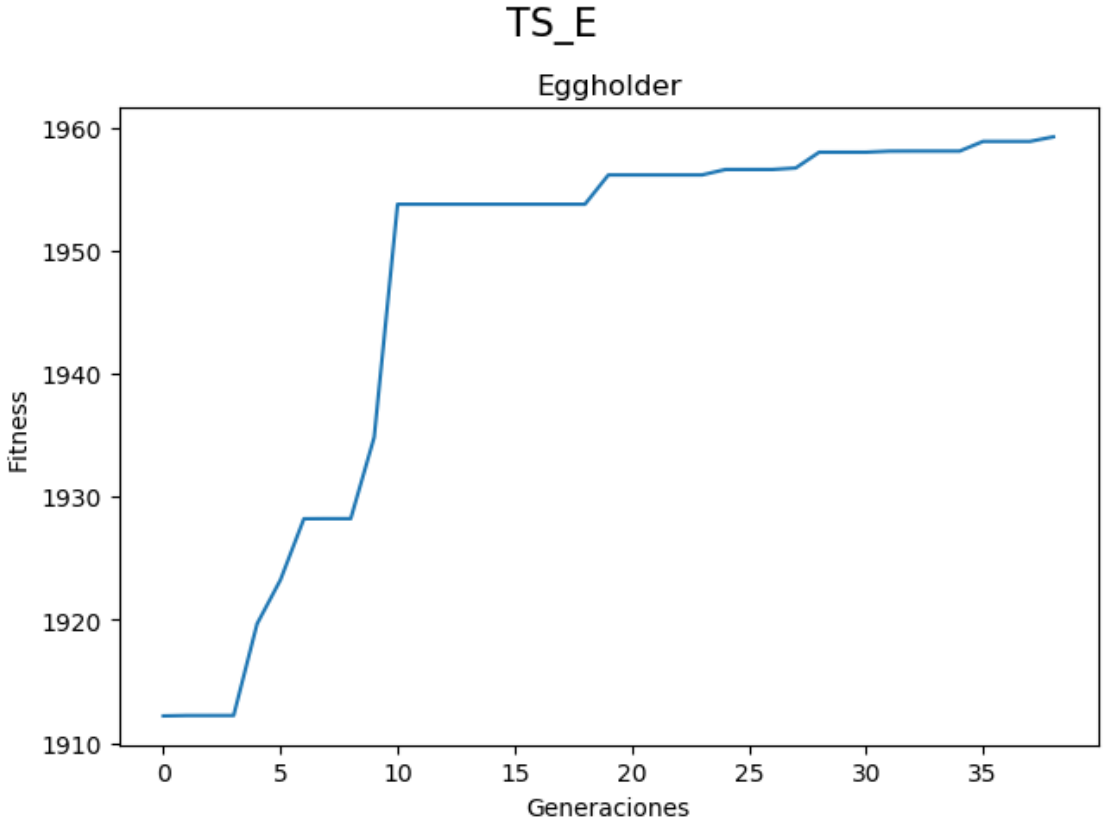


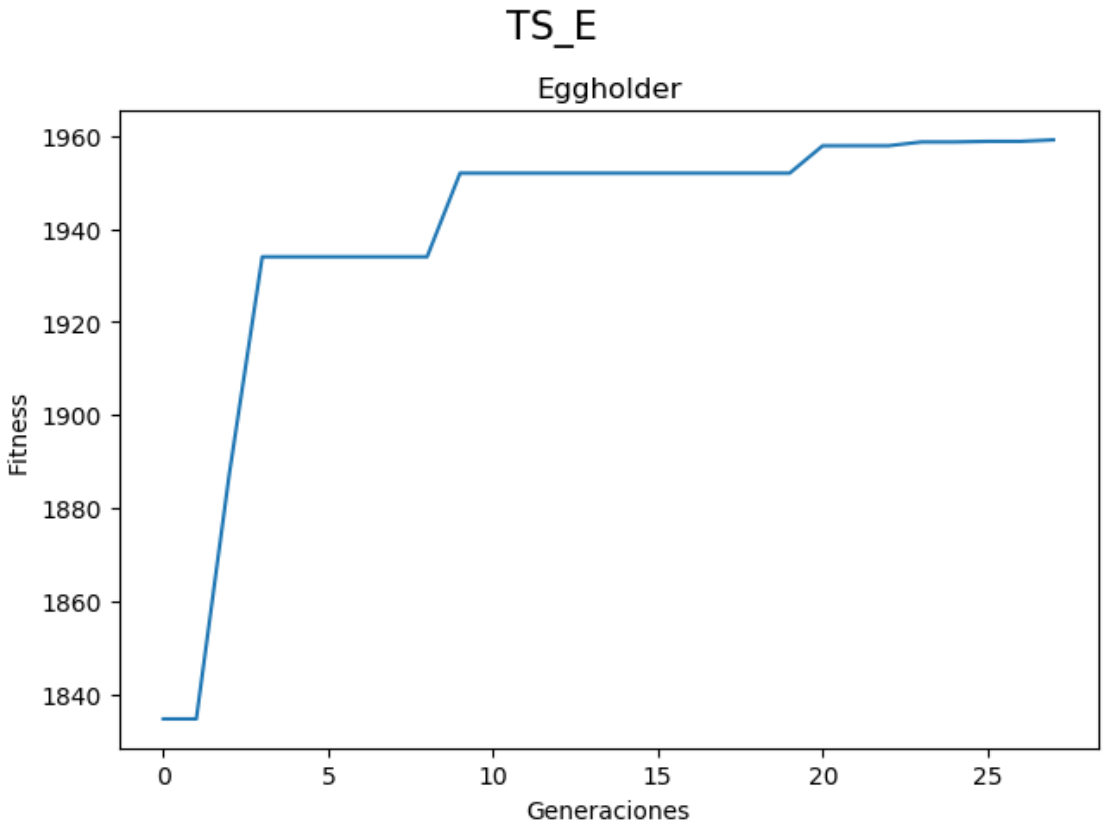
2.5.4. Eggholder

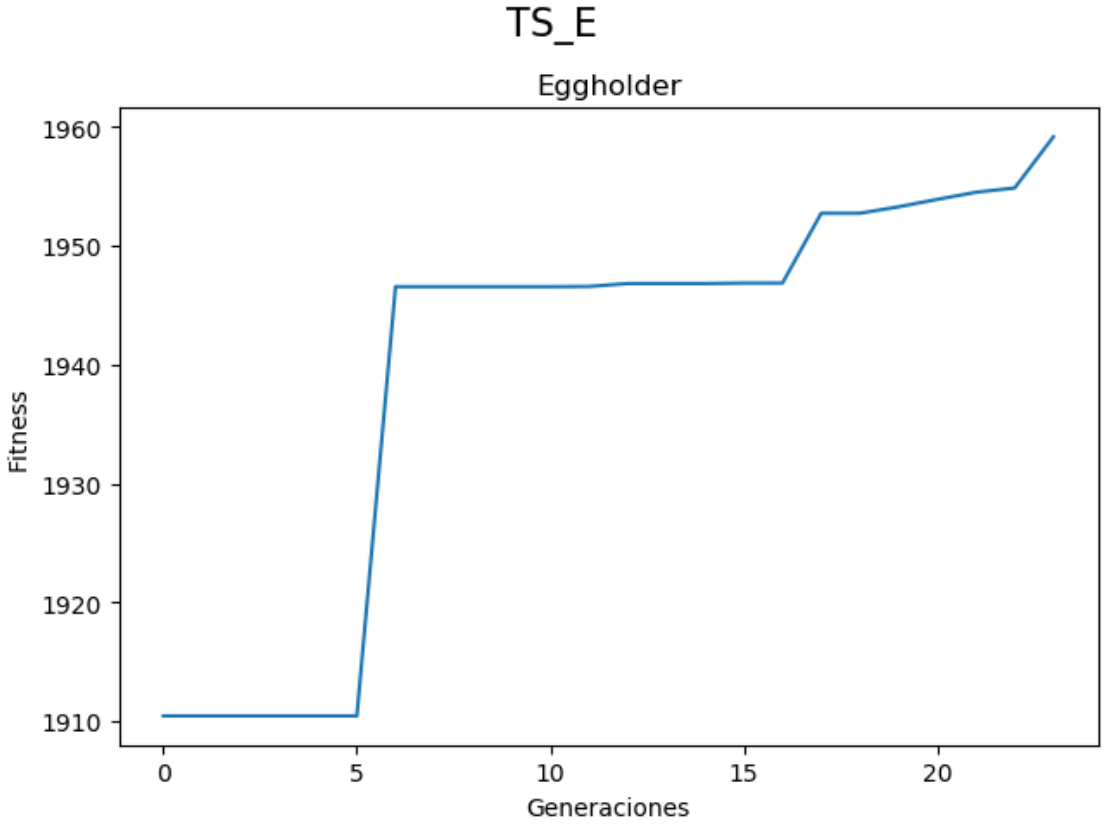
Run: 0
Selection method: TS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[511.8774414 403.89244961] [511.87939452 403.8007748]]
after: 123 generations.
Run: 1
Selection method: TS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[511.91381835 403.91320157]]
after: 39 generations.
Run: 2
Selection method: TS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[511.94232177 403.55822108] [511.91290283 403.63854335] [511.94232177

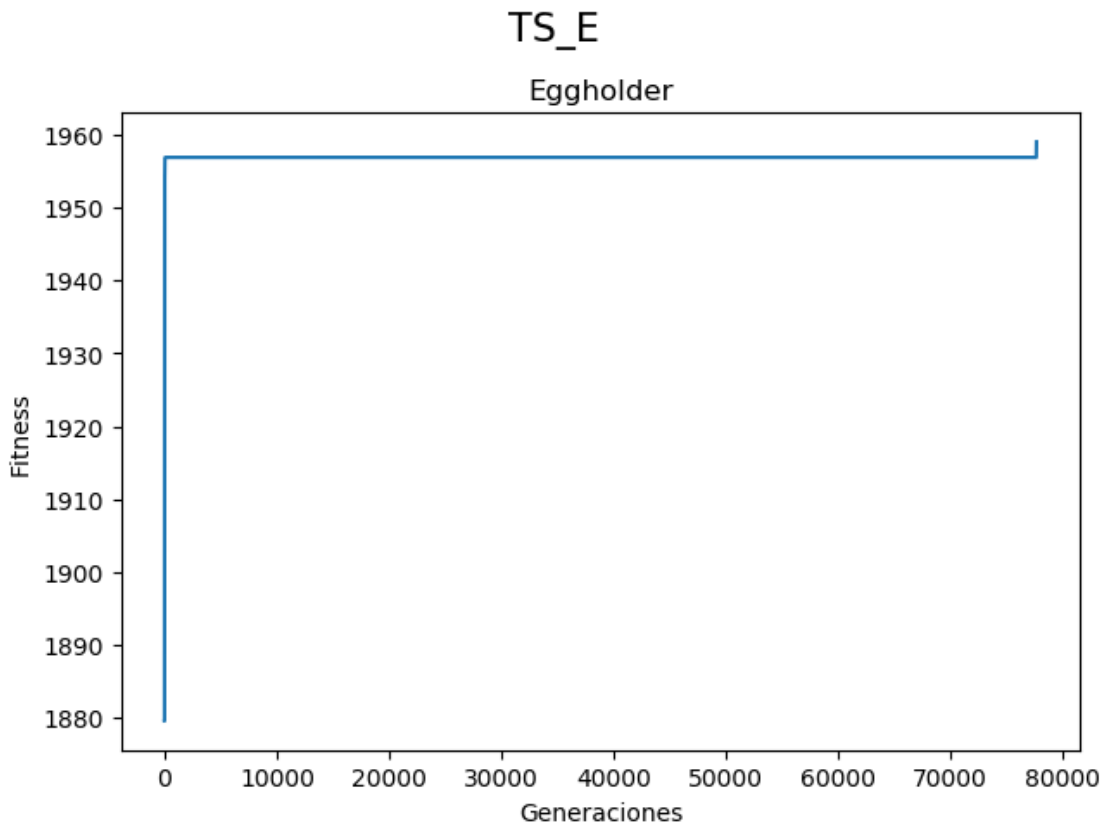
403.59667323] [511.97357178 403.64959071]] after: 28 generations.
Run: 3
Selection method: TS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[511.89685058 403.83129238]]
after: 24 generations.
Run: 4
Selection method: TS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'tournament', 'elitism': 0.1
Global minimum found: [[511.97772217 404.88268404]]
after: 77730 generations.


TS_E
Eggholder
Fitness
Generaciones
1960
1955
1950
1945
1940
1935
1930
1925
0
20
40
60
80
100
120


TS_E
Eggholder
Fitness
Generaciones
1960
1950
1940
1930
1920
1910
0
5
10
15
20
25
30
35


TS_E
Eggholder
1960
1940
1920
1900
1880
1860
1840
Fitness
0
5
10
15
20
25
Generaciones


TS_E
Eggholder
1960
1950
1940
1930
1920
1910
Fitness
0
5
10
15
20
Generaciones

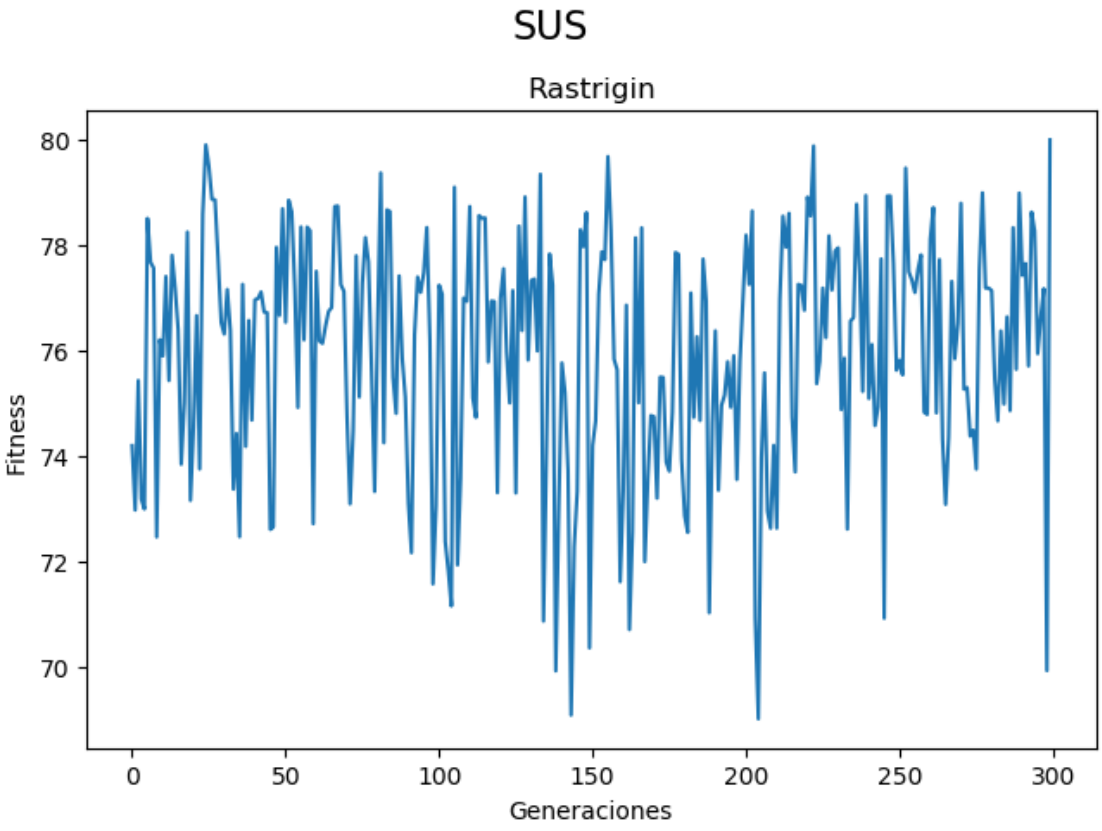


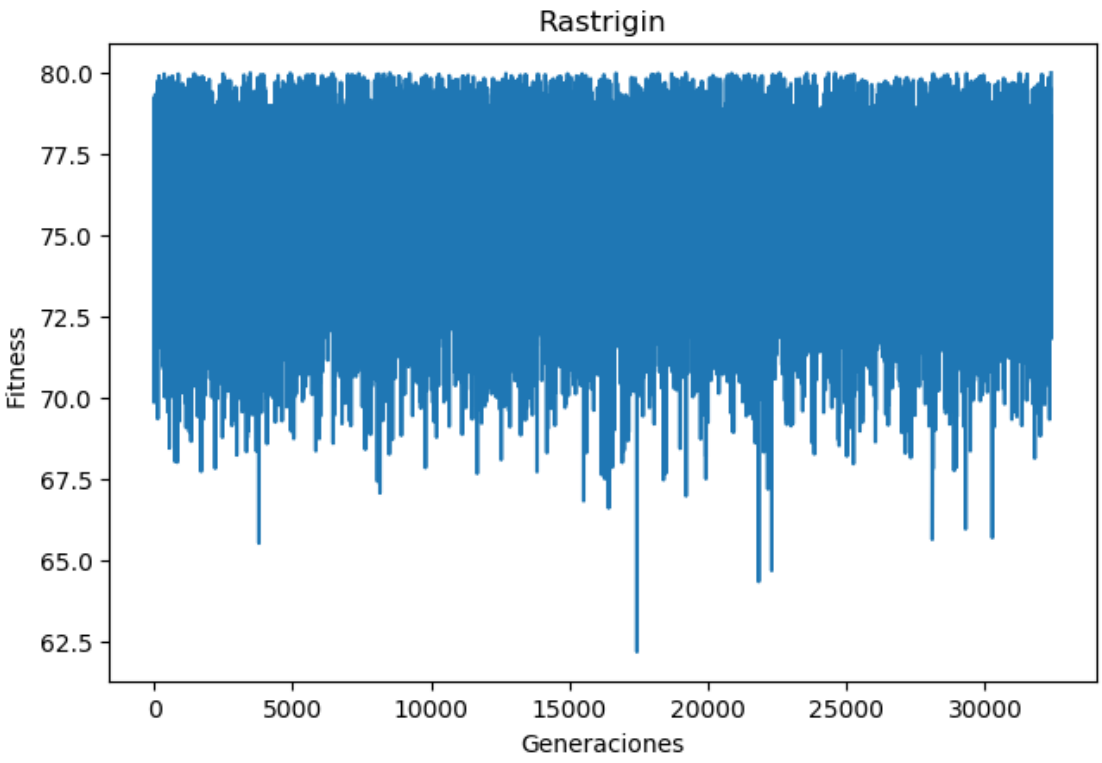
2.6. Selección por SUS, sin elitismo

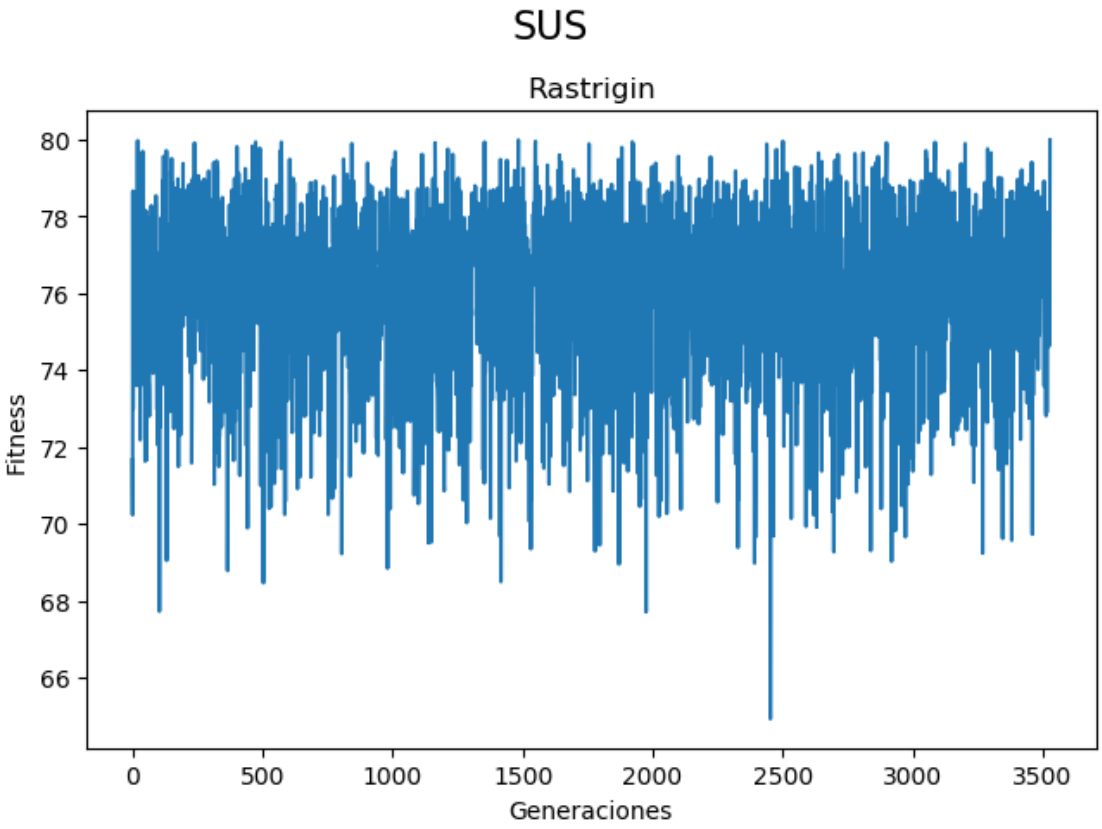
2.6.1. Rastrigin

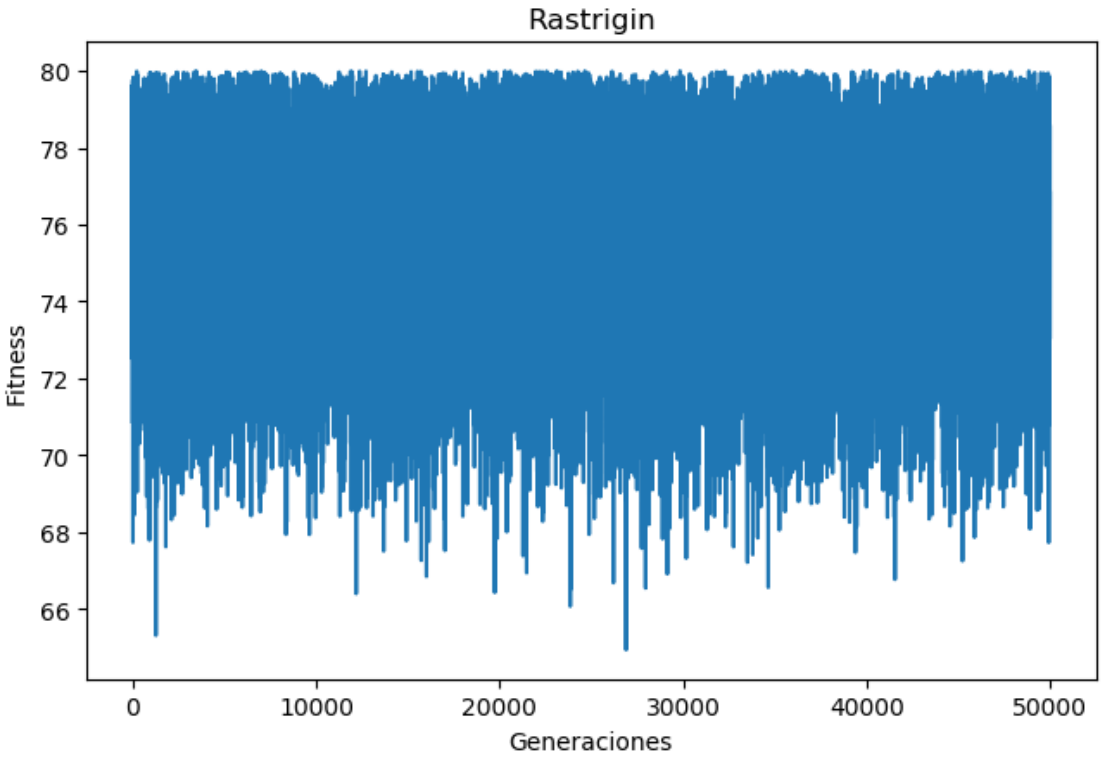
Run: 0
Selection method: SUS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[0.00097657 -0.00160157]]
after: 300 generations.
Run: 1
Selection method: SUS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[-0.00207033 0.00074219]]
after: 32391 generations.
Run: 2
Selection method: SUS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[0.00160157 -0.00113282]]
after: 3529 generations.

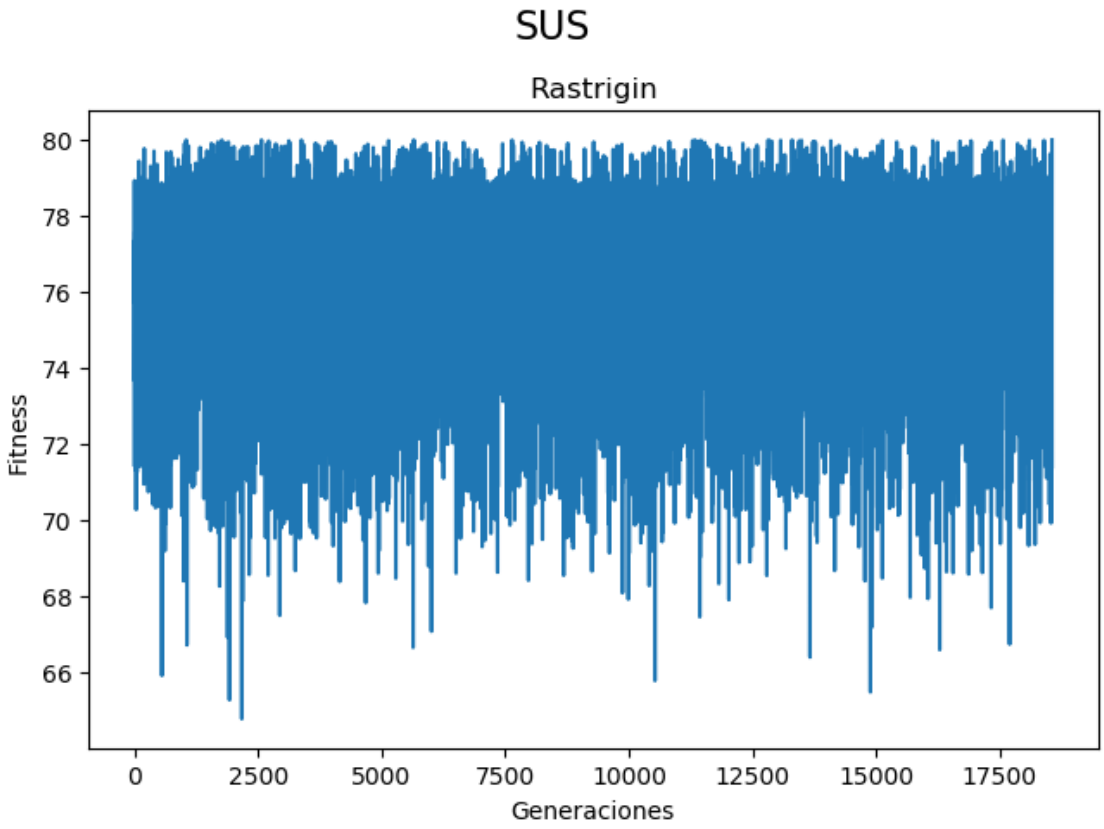
Run: 3
Selection method: SUS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[-4.43694791 4.32116578] [-3.52787848 3.48803442]]
after: 50000 generations.
Run: 4
Selection method: SUS
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[0.00160157 0.0013672]]
after: 18557 generations.


SUS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
50
100
150
200
250
300


SUS
Rastrigin
Fitness
Generaciones
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
0
5000
10000
15000
20000
25000
30000


SUS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
66
0
500
1000
1500
2000
2500
3000
3500

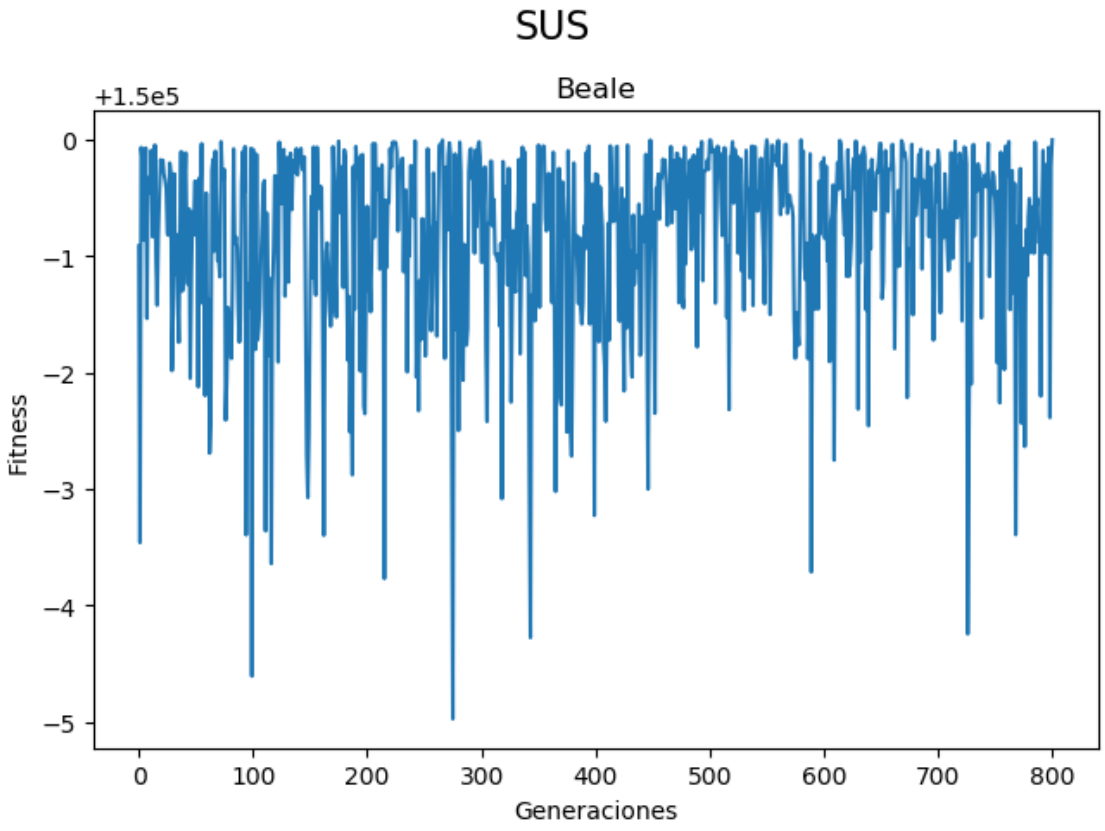

SUS
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
68
66
0
10000
20000
30000
40000
50000

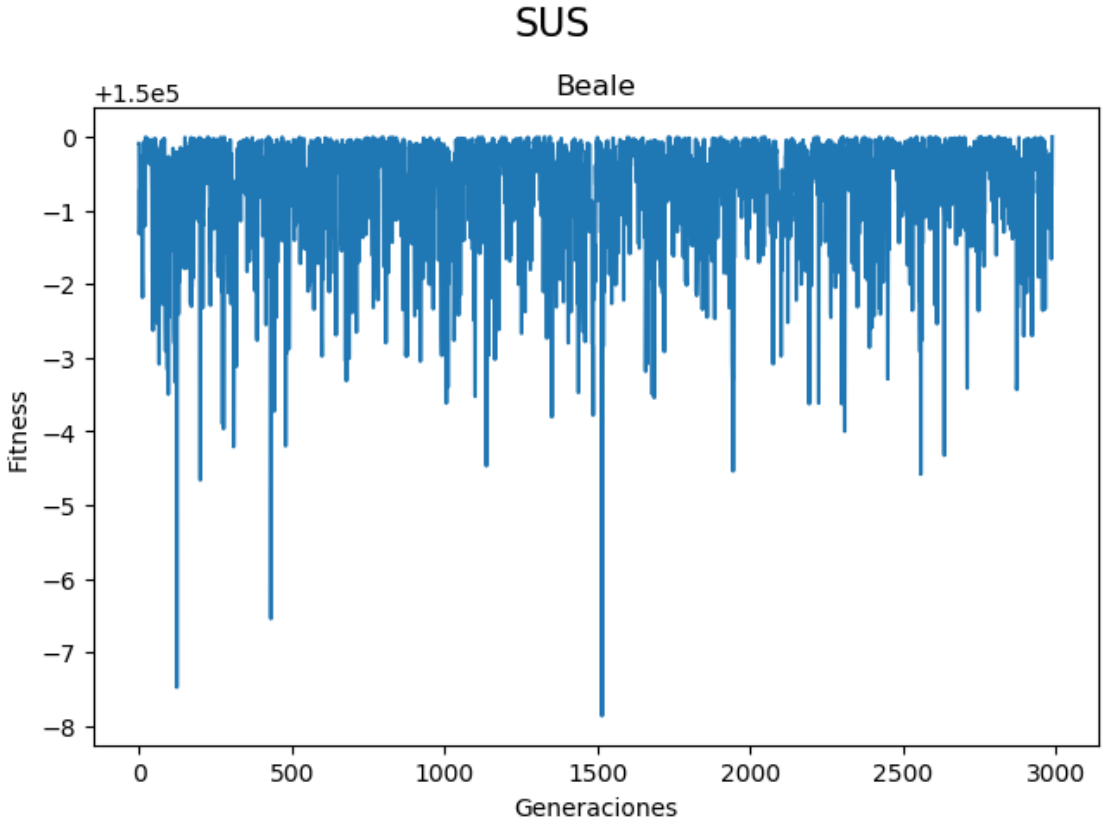


2.6.2. Beale

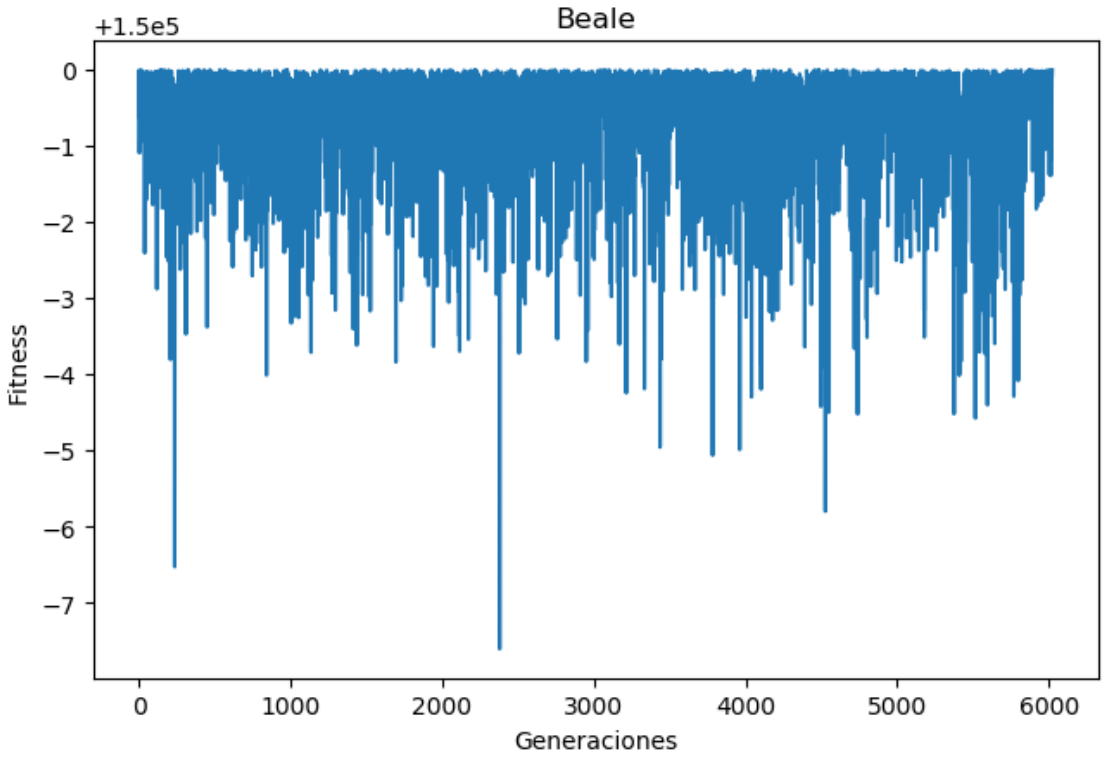
Run: 0
Selection method: SUS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[2.99733351 0.49826811]]
after: 801 generations.
Run: 1
Selection method: SUS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.01216516 0.503212]]
after: 2989 generations.
Run: 2
Selection method: SUS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[2.9961662 0.50060273]]

after: 6024 generations.
Run: 3
Selection method: SUS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.01559842 0.50417331]]
after: 6307 generations.
Run: 4
Selection method: SUS
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.0232889 0.50602727]]
after: 9823 generations.

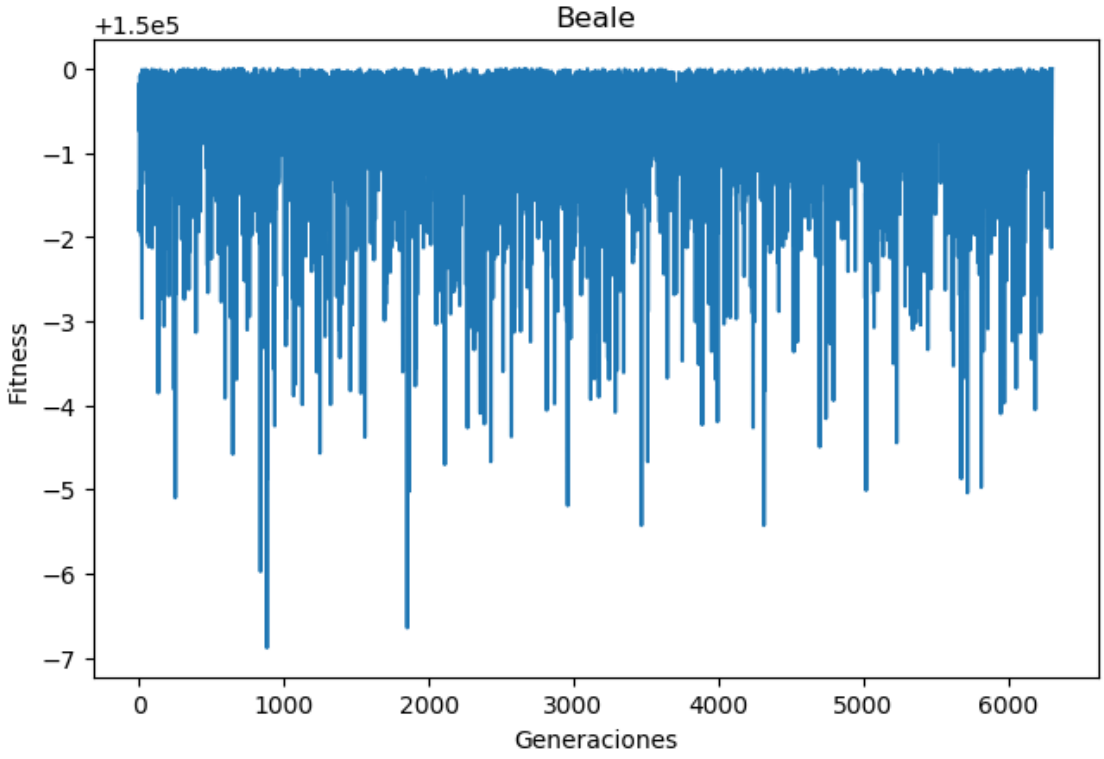

SUS
Beale
+1.5e5
0
−1
−2
−3
−4
−5
Fitness
0
100
200
300
400
500
600
700
800
Generaciones

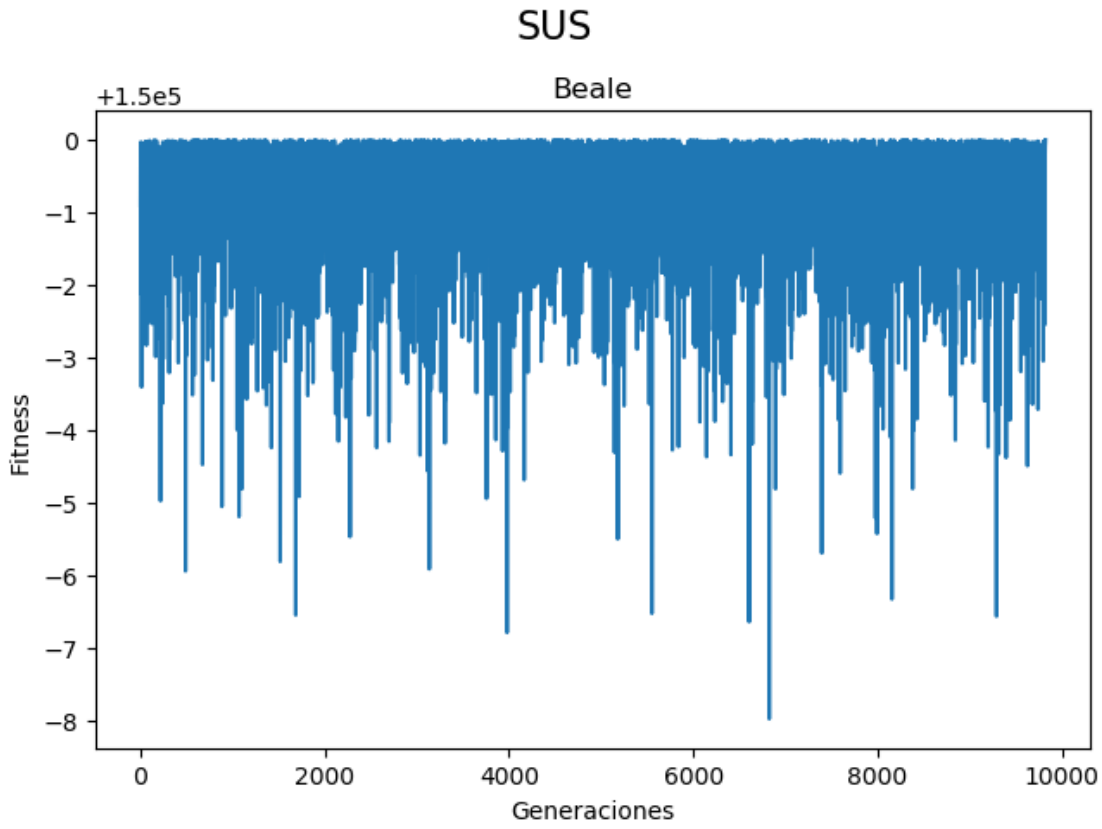

SUS
Beale
+1.5e5
0
−1
−2
−3
−4
−5
−6
−7
−8
Fitness
0
500
1000
1500
2000
2500
3000
Generaciones

SUS


Beale
+1.5e5
0
−1
−2
−3
−4
−5
−6
−7
Fitness
0
1000
2000
3000
4000
5000
6000
Generaciones

SUS

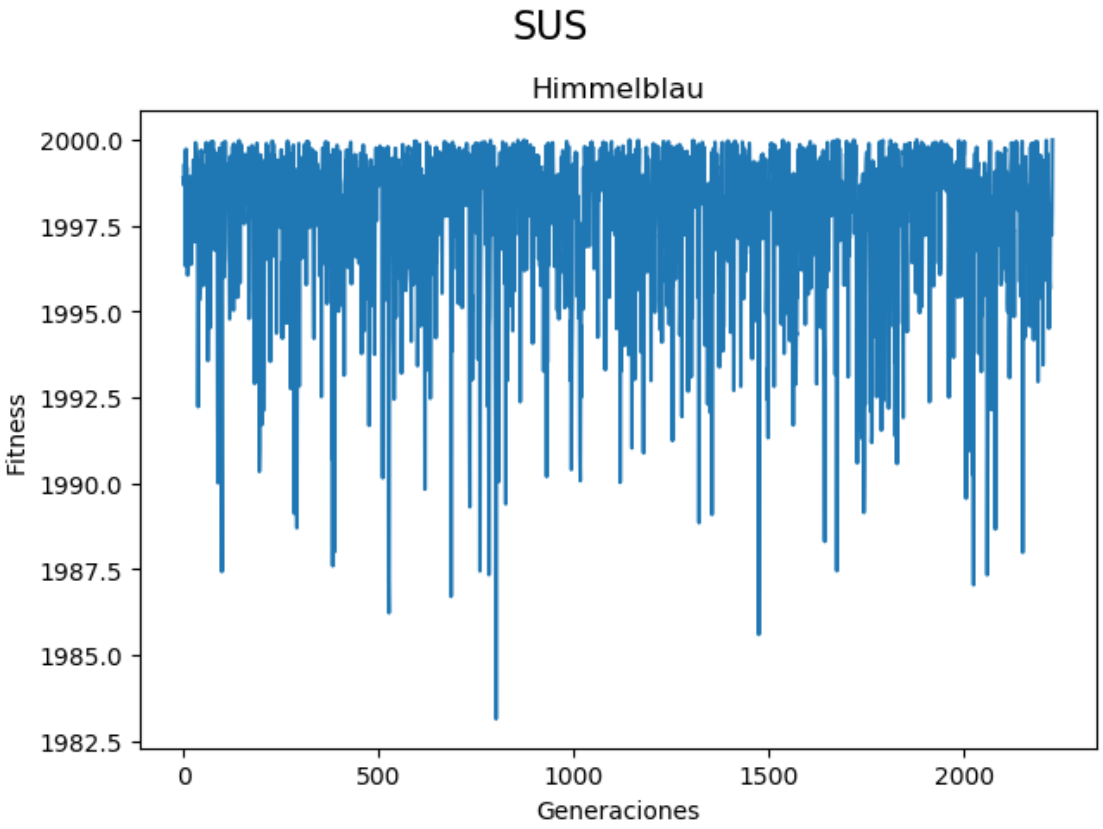

Beale
+1.5e5
0
−1
−2
−3
−4
−5
−6
−7
Fitness
0
1000
2000
3000
4000
5000
6000
Generaciones

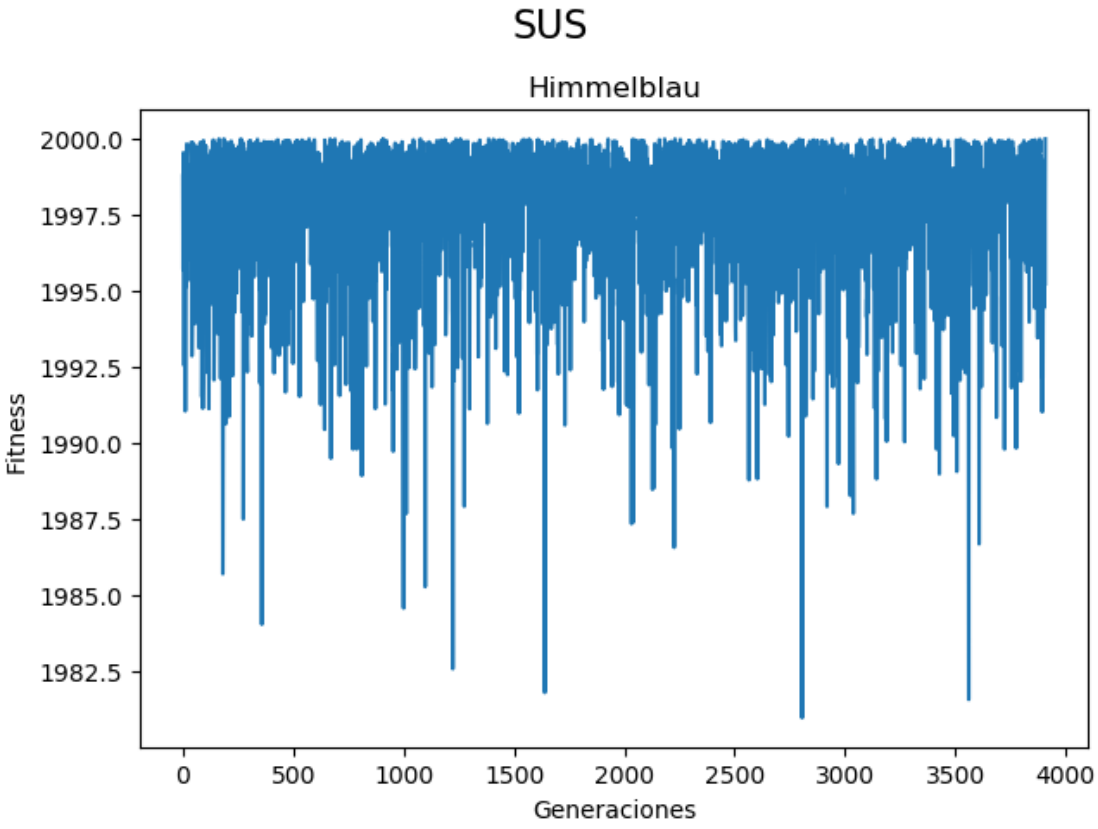


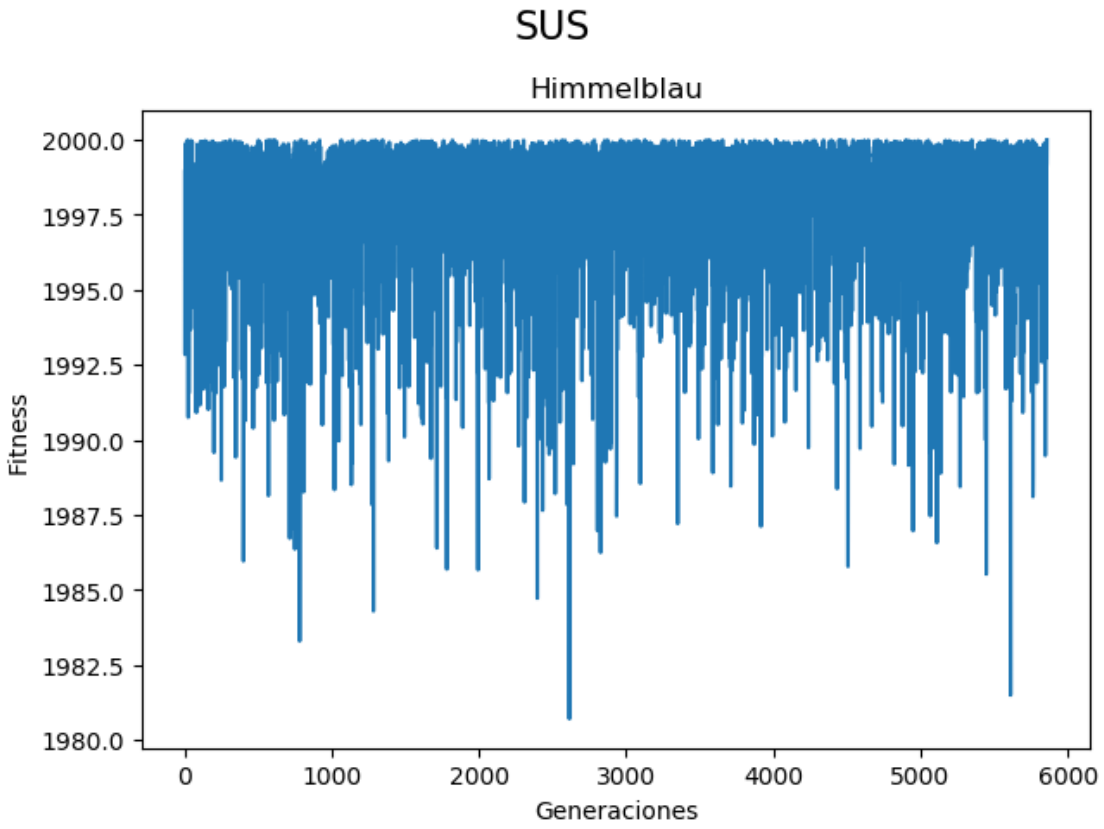
2.6.3. Himmelblau

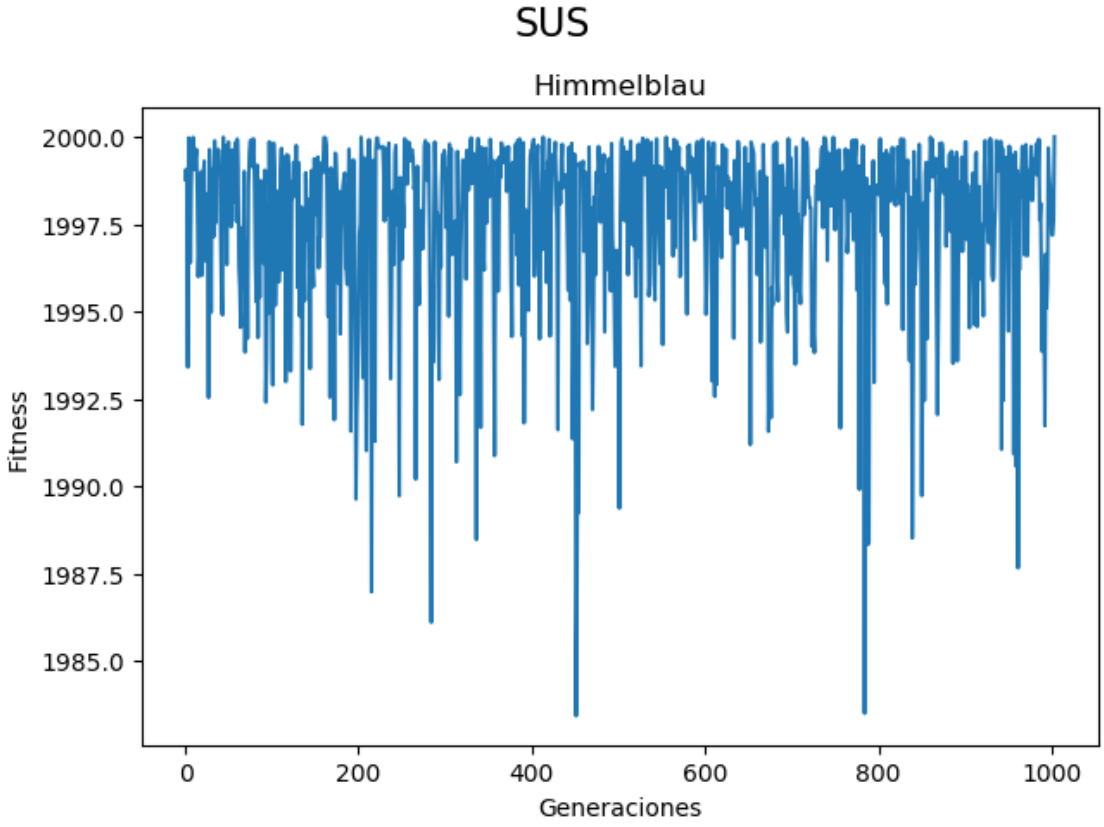
Run: 0
Selection method: SUS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.00093079 1.99796294]]
after: 2227 generations.
Run: 1
Selection method: SUS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.58298174 -1.85132485]]
after: 3909 generations.
Run: 2
Selection method: SUS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[-2.80622716 3.13070778]]

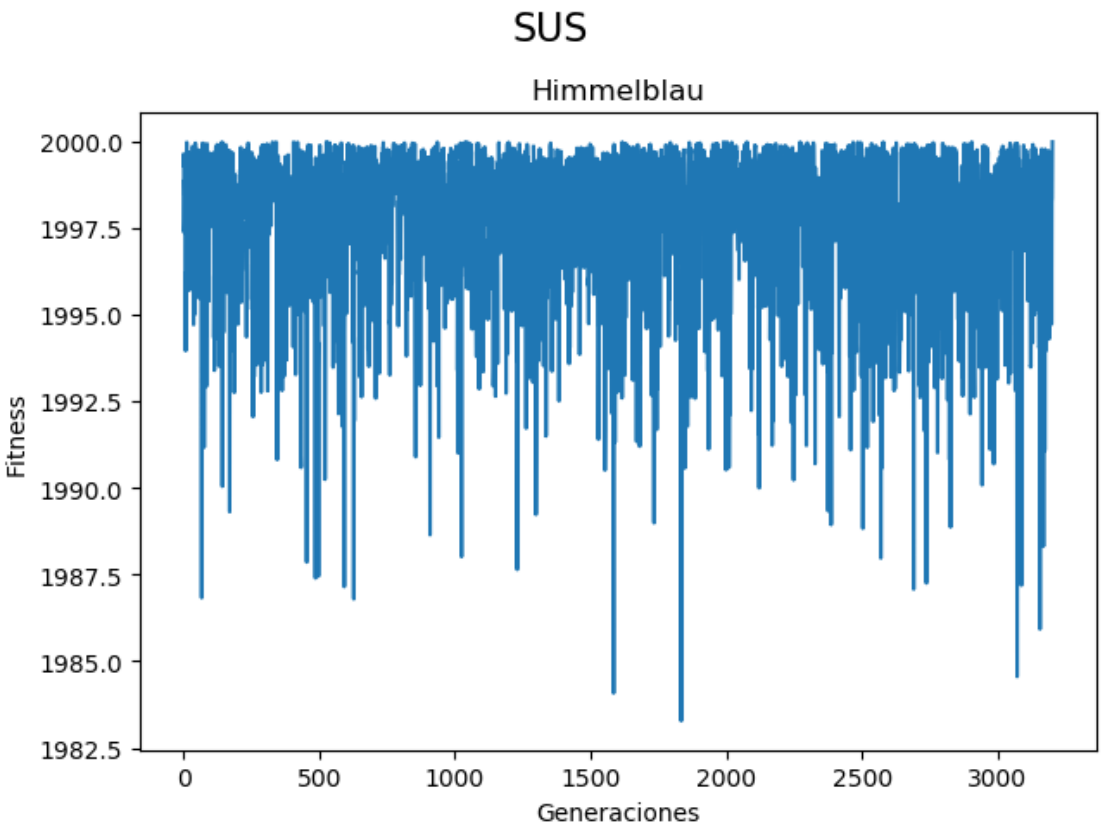
after: 5861 generations.
Run: 3
Selection method: SUS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.0049744 1.99361415]]
after: 1004 generations.
Run: 4
Selection method: SUS
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus'
Global minimum found: [[3.00352481 1.9984207]]
after: 3202 generations.


SUS
Himmelblau
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
Fitness
0
500
1000
1500
2000
Generaciones


SUS
Himmelblau
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
Fitness
0
500
1000
1500
2000
2500
3000
3500
4000
Generaciones


SUS
Himmelblau
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
1982.5
1980.0
Fitness
0
1000
2000
3000
4000
5000
6000
Generaciones

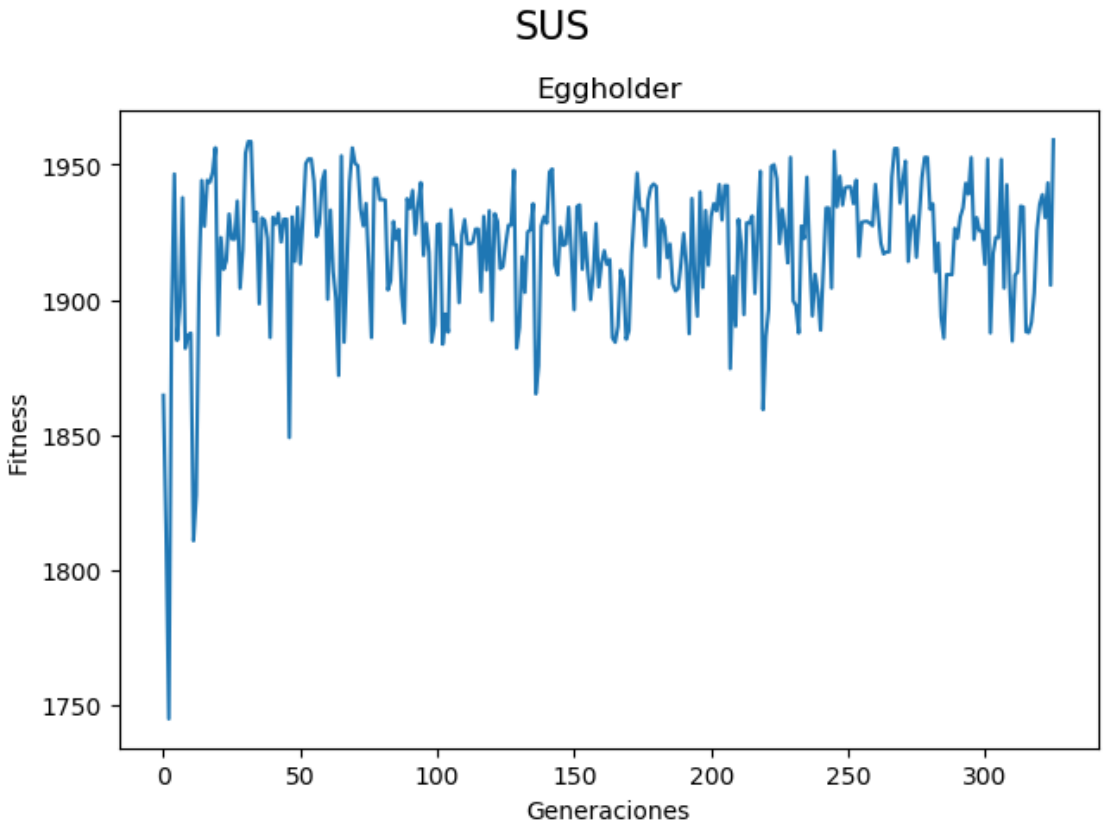

SUS
Himmelblau
2000.0
1997.5
1995.0
1992.5
1990.0
1987.5
1985.0
Fitness
0
200
400
600
800
1000
Generaciones

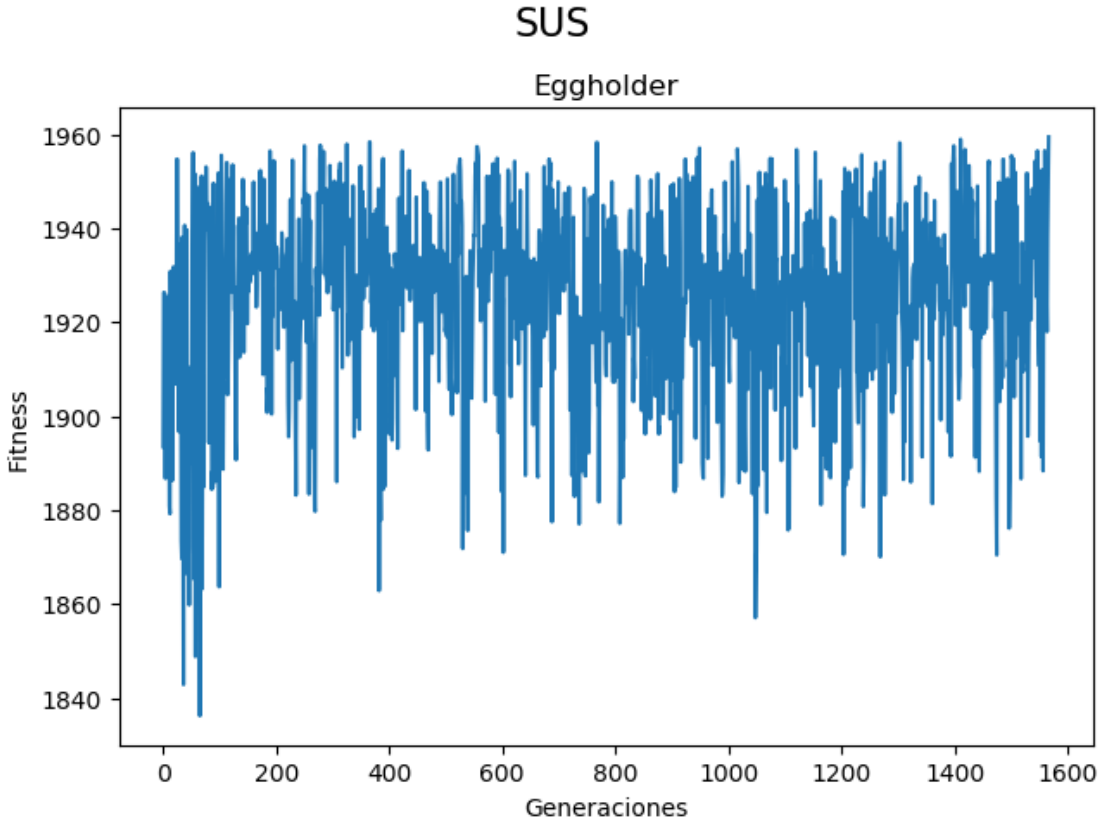


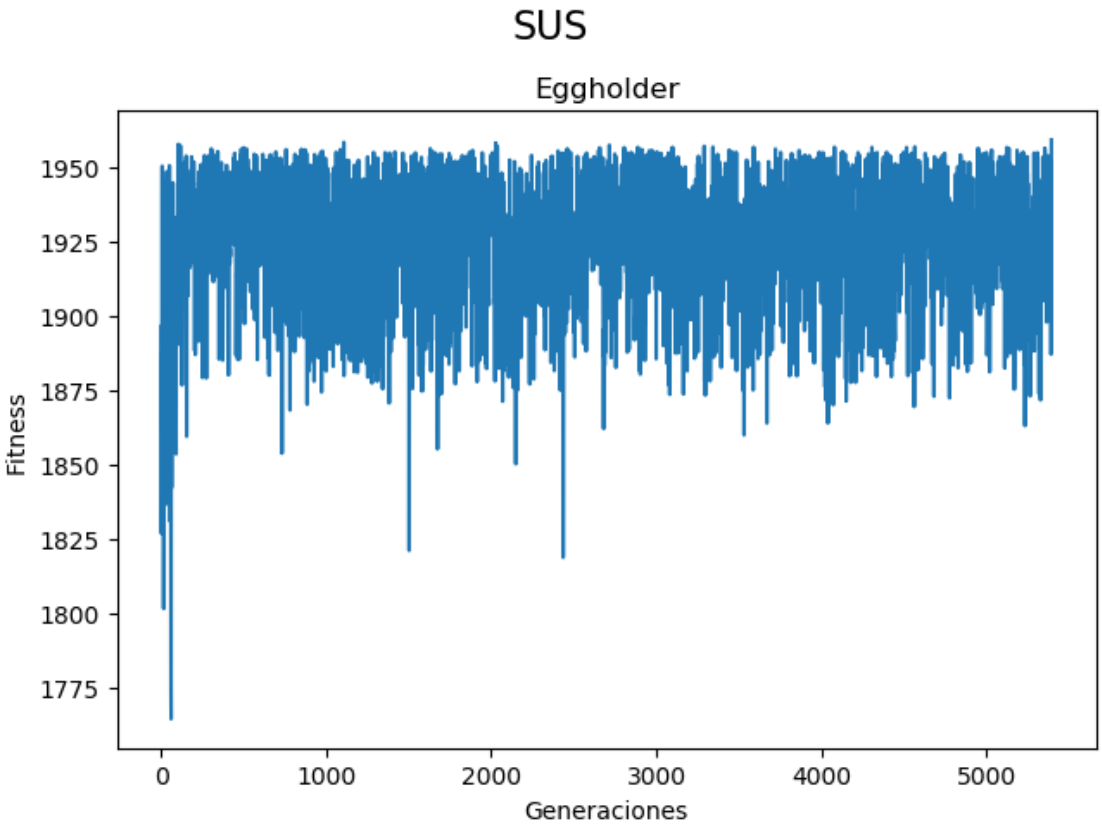
2.6.4. Eggholder

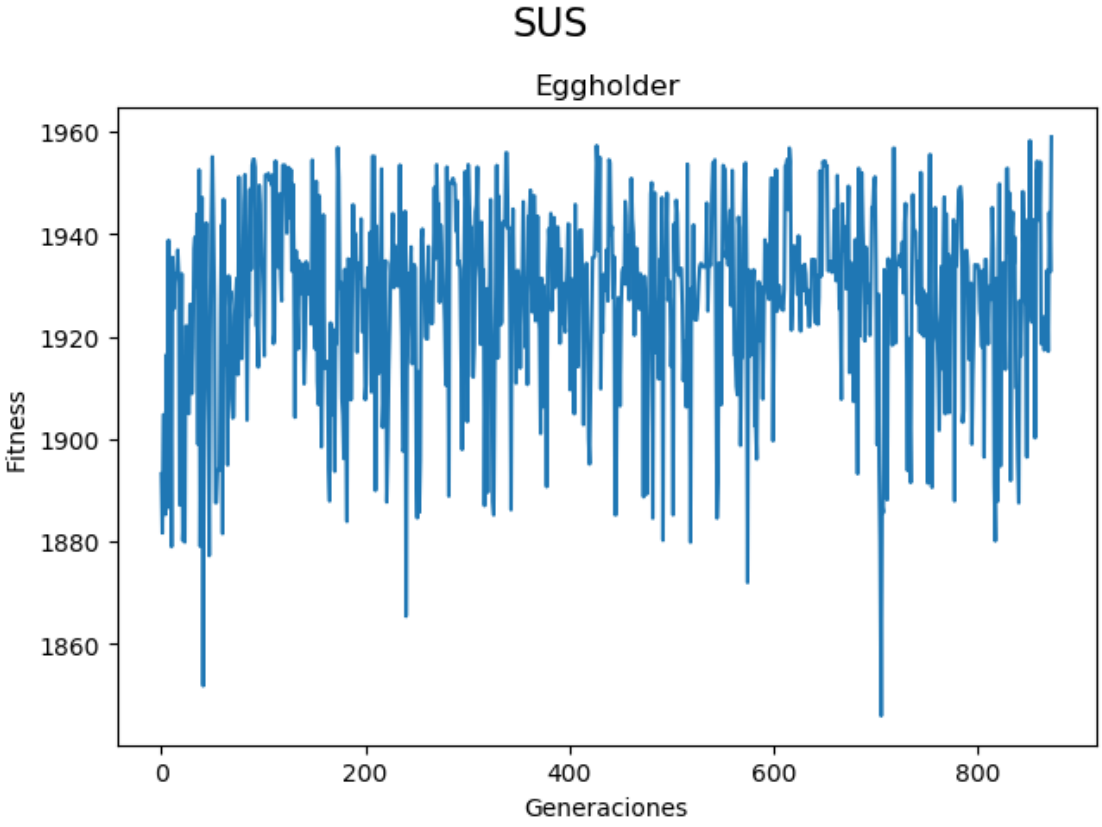
Run: 0
Selection method: SUS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000, 'selection': 'sus'
Global minimum found: [[511.99884033 403.62554286]]
after: 326 generations.
Run: 1
Selection method: SUS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000, 'selection': 'sus'
Global minimum found: [[511.99255371 404.54564789]]
after: 1568 generations.
Run: 2
Selection method: SUS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000, 'selection': 'sus'
Global minimum found: [[511.98089599 403.65465663]]

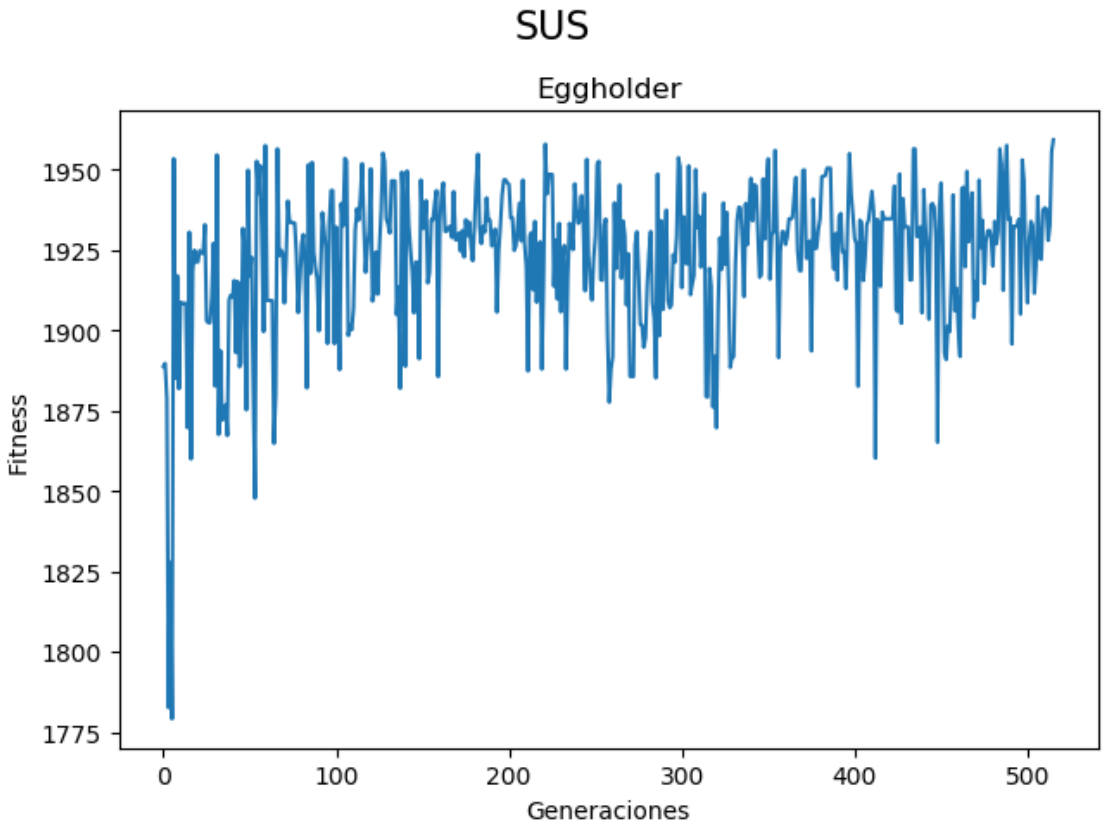
after: 5397 generations.
Run: 3
Selection method: SUS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000, 'selection': 'sus'
Global minimum found: [[511.8128662 404.07457853]]
after: 874 generations.
Run: 4
Selection method: SUS
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.043478260869565216, 'max_iter': 100000, 'selection': 'sus'
Global minimum found: [[511.92712402 404.30413176]]
after: 516 generations.


SUS
Eggholder
Fitness
Generaciones
1950
1900
1850
1800
1750
0
50
100
150
200
250
300


SUS
Eggholder
Fitness
Generaciones
1960
1940
1920
1900
1880
1860
1840
0
200
400
600
800
1000
1200
1400
1600


SUS
Eggholder
Fitness
1950
1925
1900
1875
1850
1825
1800
1775
0
1000
2000
3000
4000
5000
Generaciones


SUS
Eggholder
Fitness
1960
1940
1920
1900
1880
1860
0
200
400
600
800
Generaciones

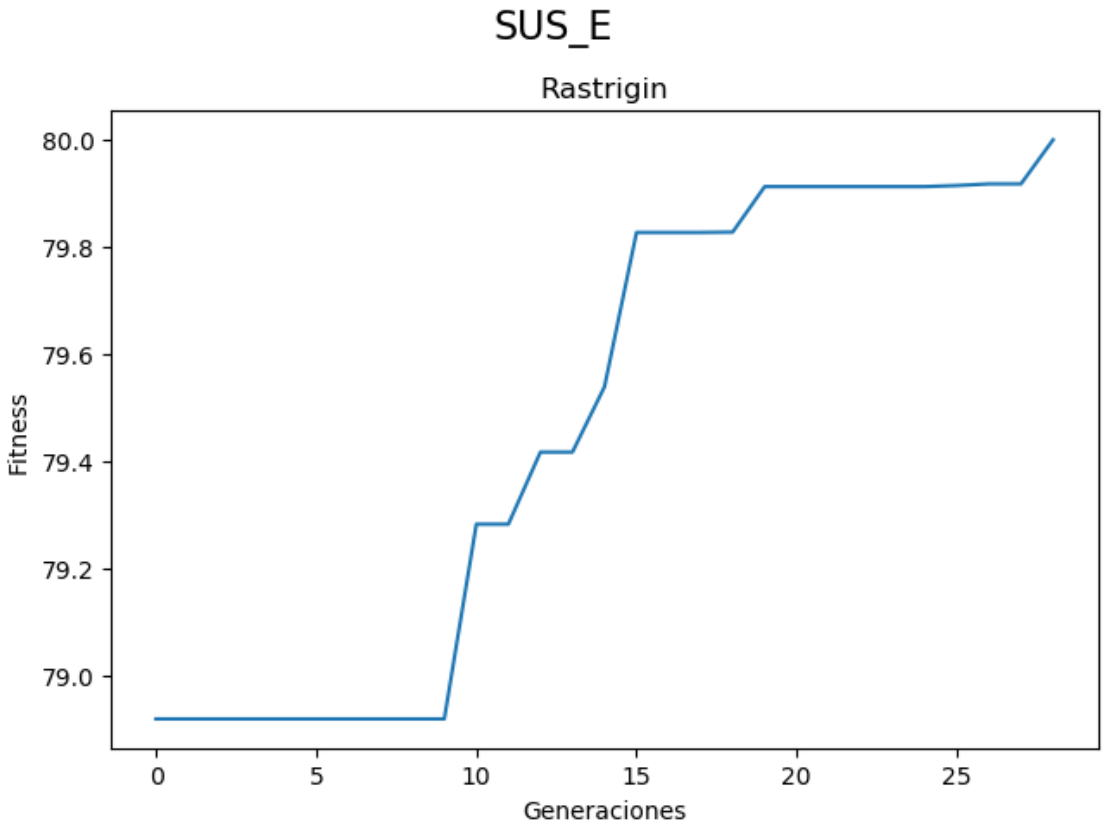


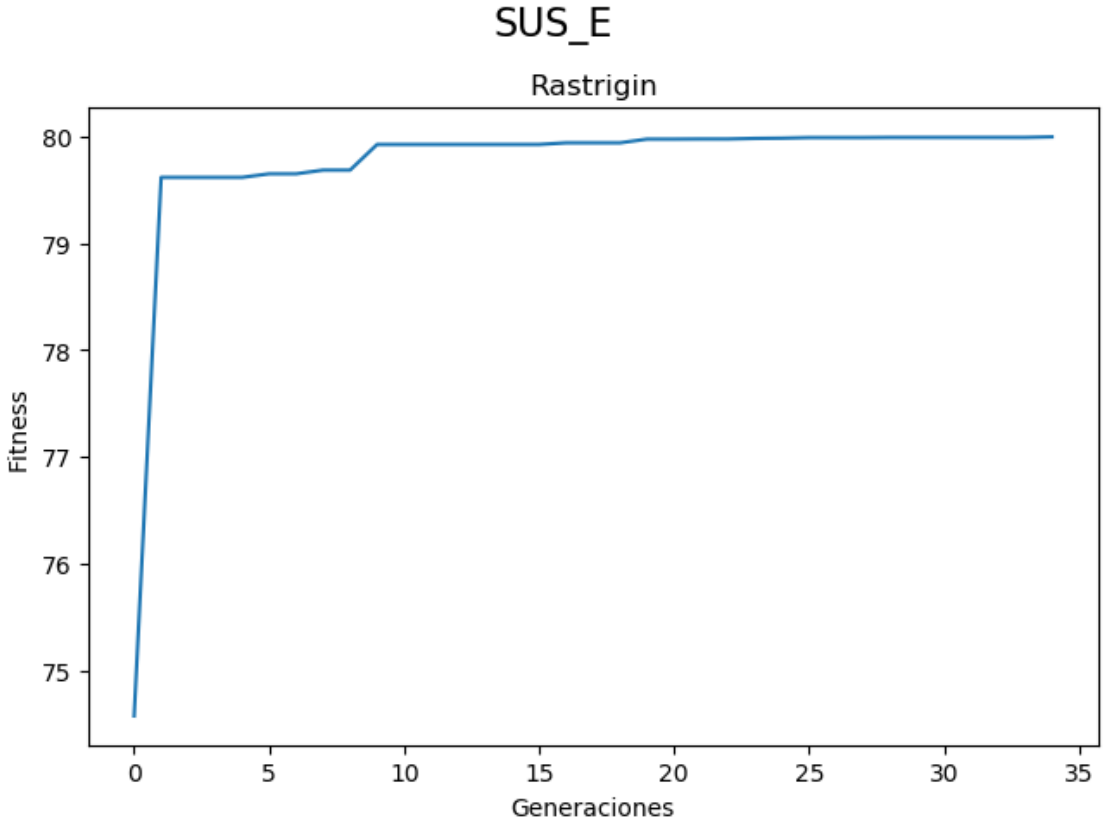
2.7. Selección por SUS, con elitismo

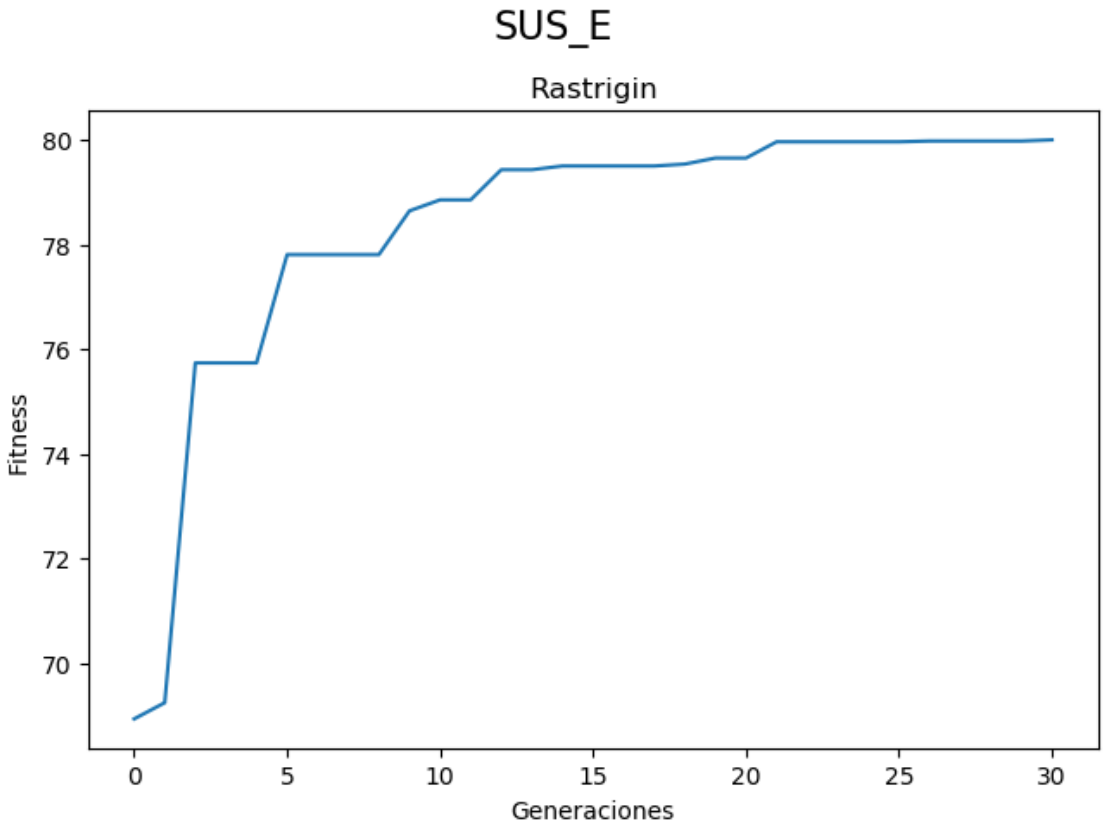
2.7.1. Rastrigin

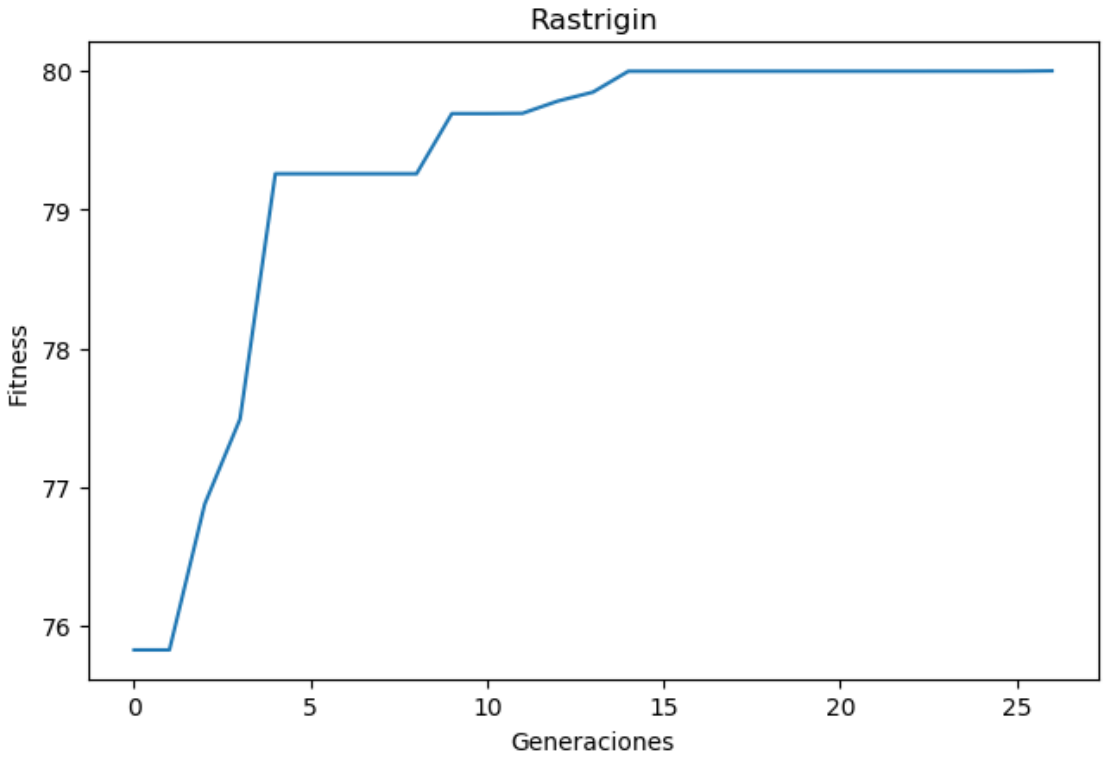
Run: 0
Selection method: SUS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[-0.00089844 0.00074219] [-0.00027344 0.00027344]]
after: 29 generations.
Run: 1
Selection method: SUS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[0.00042969 -0.0013672]]
after: 35 generations.
Run: 2
Selection method: SUS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus',

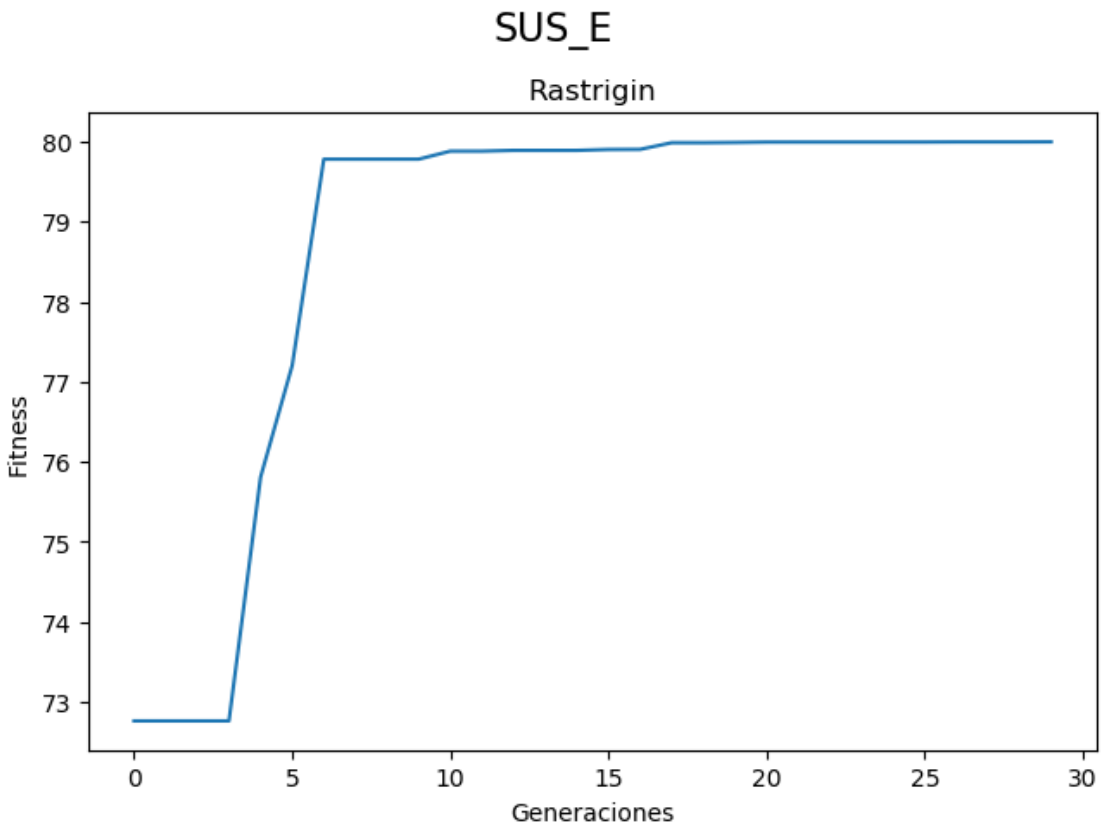
'elitism': 0.2
Global minimum found: [[-0.00027344 0.0019922]]
after: 31 generations.
Run: 3
Selection method: SUS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[-0.00152345 0.00027344]]
after: 27 generations.
Run: 4
Selection method: SUS_E
Function to optimize: Rastrigin
Parameters: 'n_individuals': 100, 'pc': 0.9, 'pm': 0.1, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[-0.00183595 0.00050782]]
after: 30 generations.


SUS_E
Rastrigin
80.0
79.8
79.6
79.4
79.2
79.0
Fitness
0
5
10
15
20
25
Generaciones


SUS_E
Rastrigin
80
79
78
77
76
75
Fitness
0
5
10
15
20
25
30
35
Generaciones


SUS_E
Rastrigin
Fitness
Generaciones
80
78
76
74
72
70
0
5
10
15
20
25
30

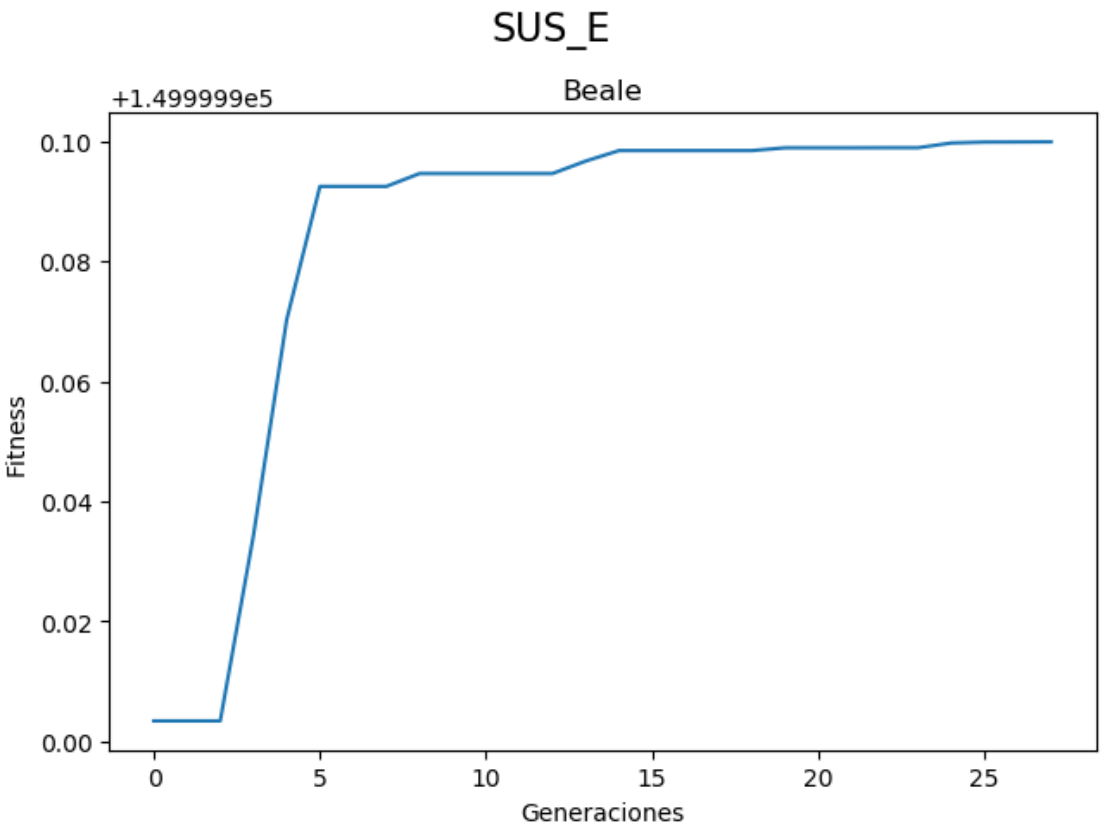

SUS_E
Rastrigin
Fitness
Generaciones
80
79
78
77
76
0
5
10
15
20
25

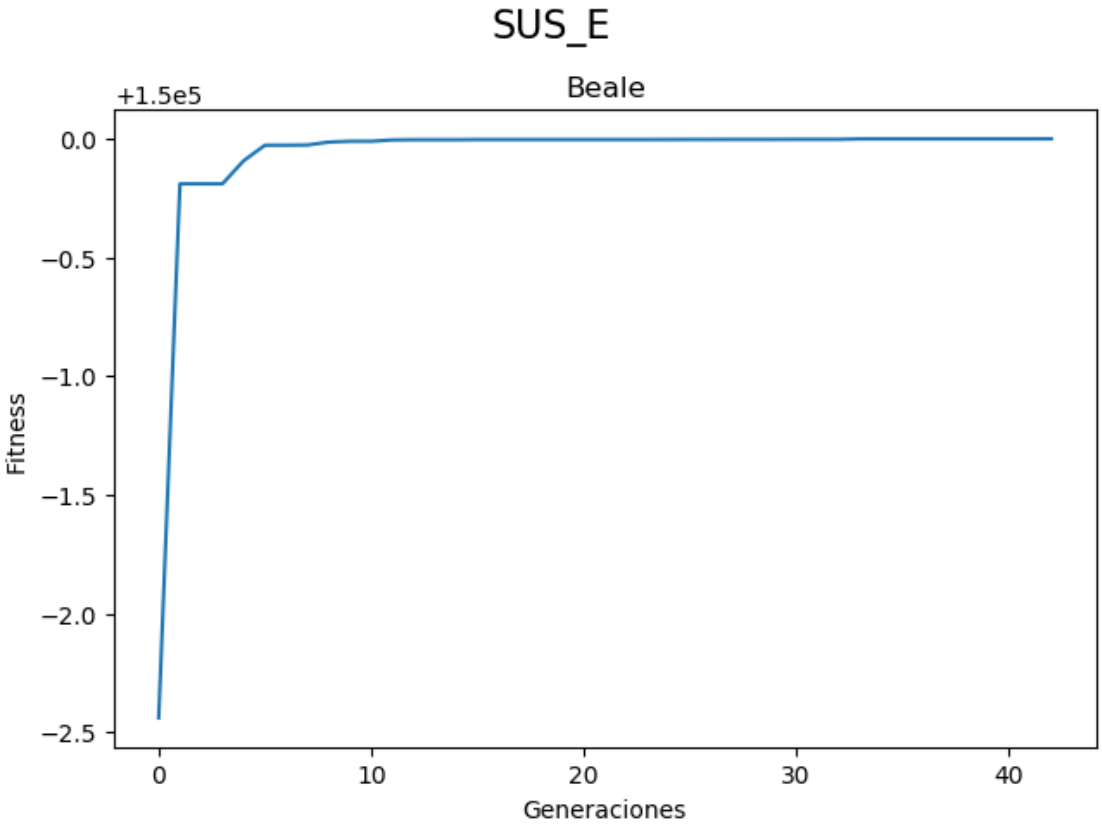


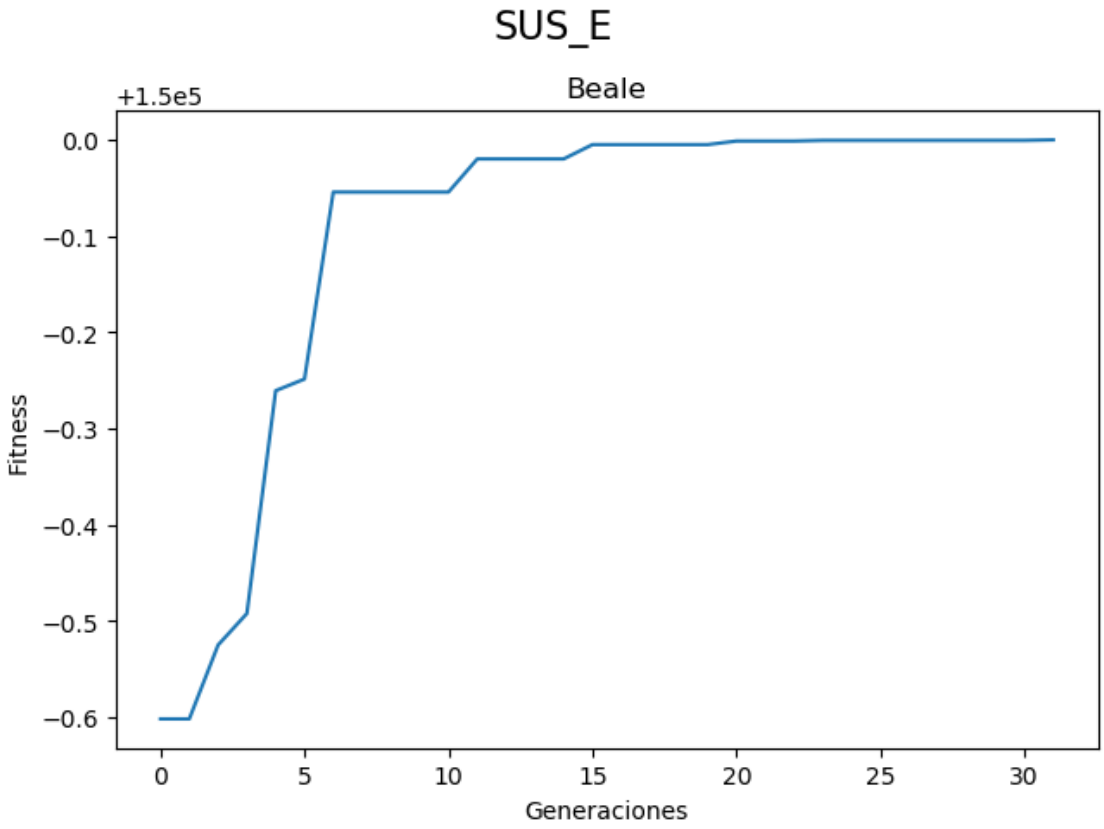
2.7.2. Beale

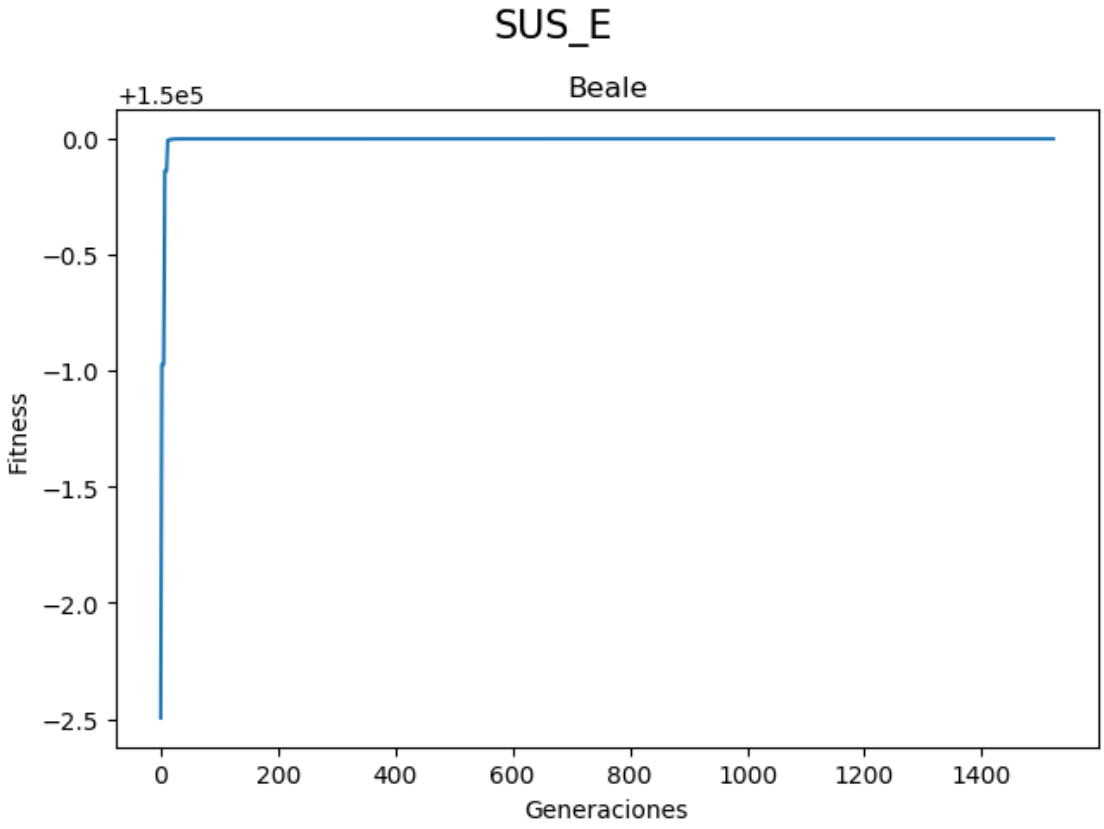
Run: 0
Selection method: SUS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.00152208 0.50218202]]
after: 28 generations.
Run: 1
Selection method: SUS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[2.99884414 0.49895476]]
after: 43 generations.
Run: 2
Selection method: SUS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[2.98071656 0.49490353]]

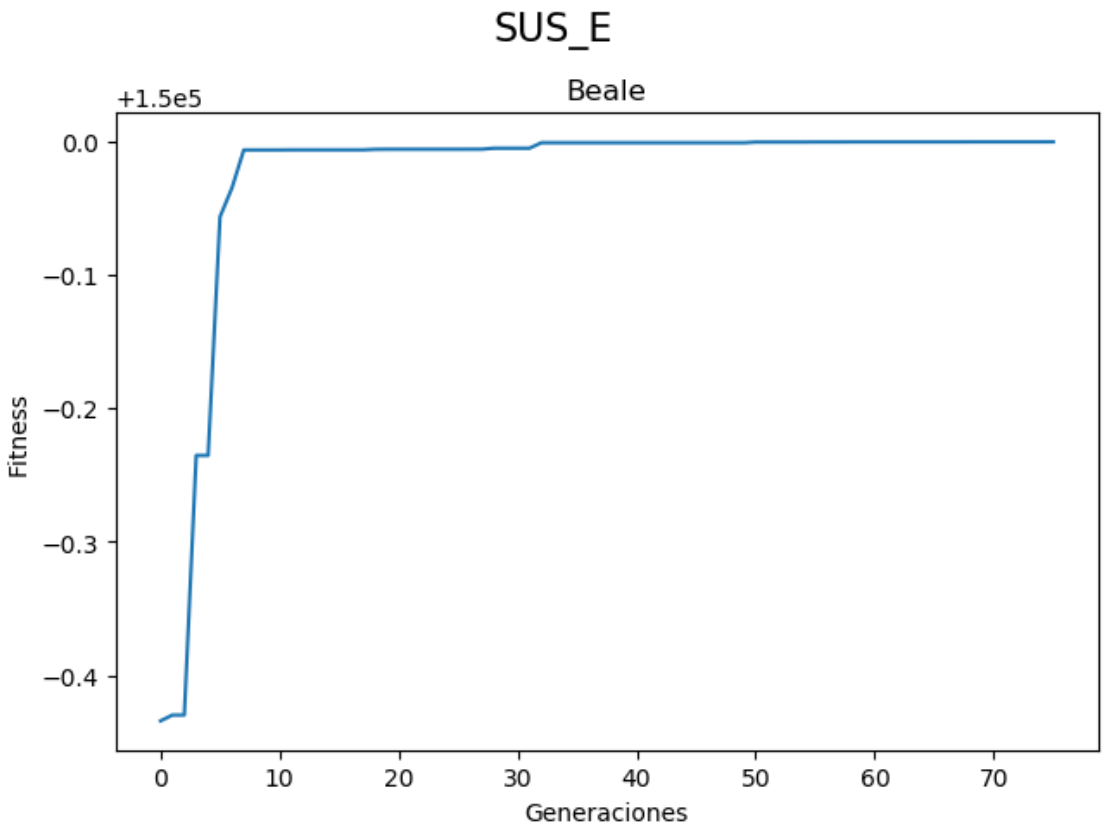
after: 32 generations.
Run: 3
Selection method: SUS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.00722128 0.50094605]]
after: 1523 generations.
Run: 4
Selection method: SUS_E
Function to optimize: Beale
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.02411289 0.50547795]]
after: 76 generations.


SUS_E
Beale
+1.499999e5
0.10
0.08
0.06
0.04
0.02
0.00
Fitness
0
5
10
15
20
25
Generaciones


SUS_E
Beale
+1.5e5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Fitness
0
10
20
30
40
Generaciones


SUS_E
Beale
+1.5e5
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
Fitness
0
5
10
15
20
25
30
Generaciones

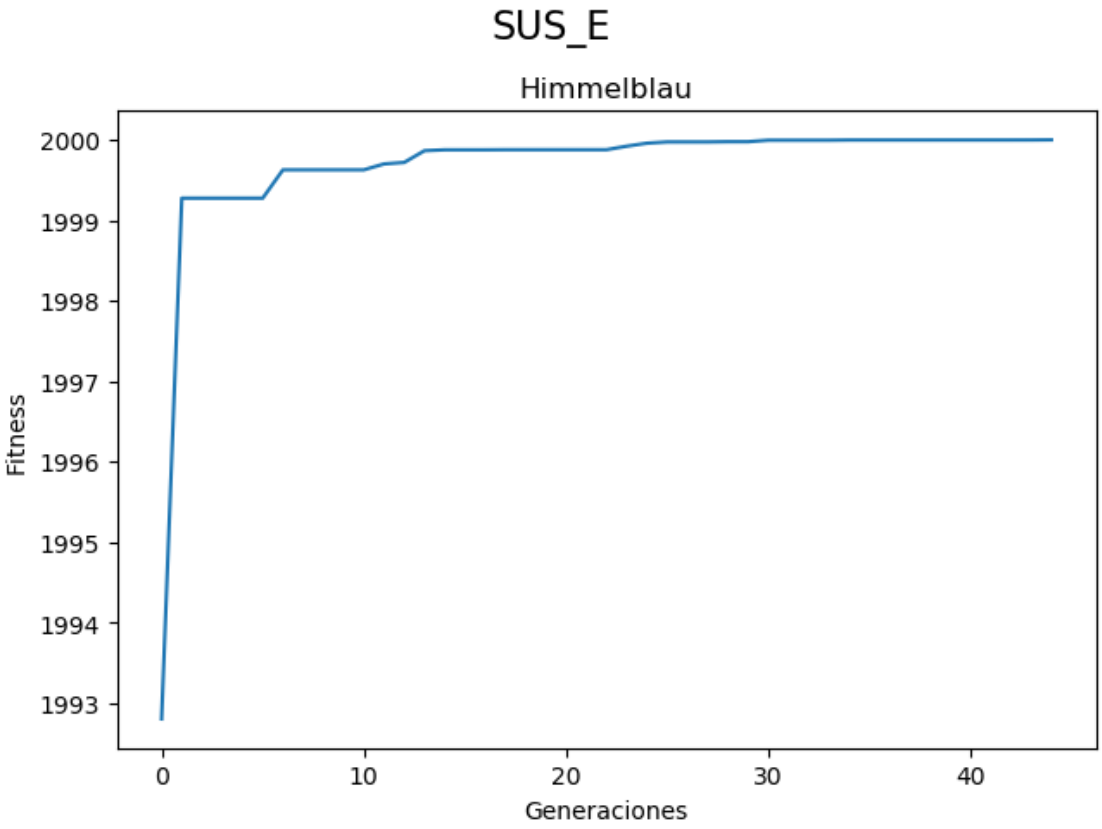

SUS_E
Beale
+1.5e5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Fitness
0
200
400
600
800
1000
1200
1400
Generaciones

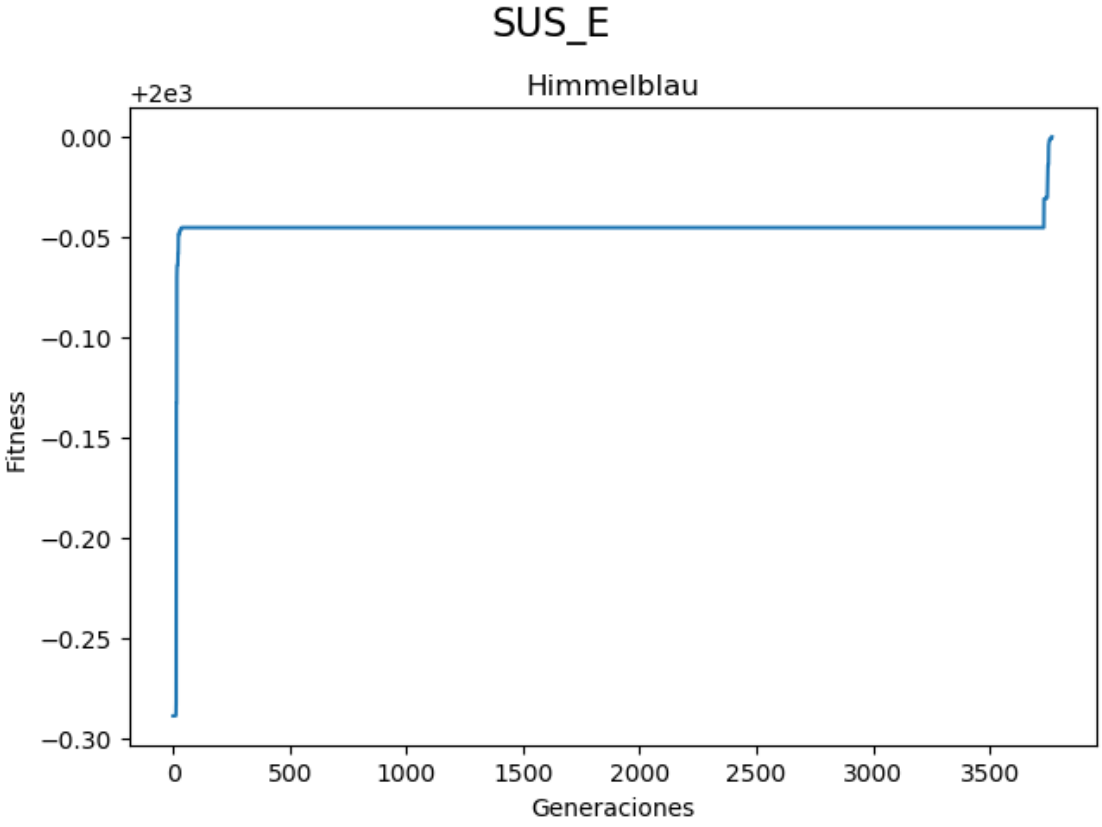


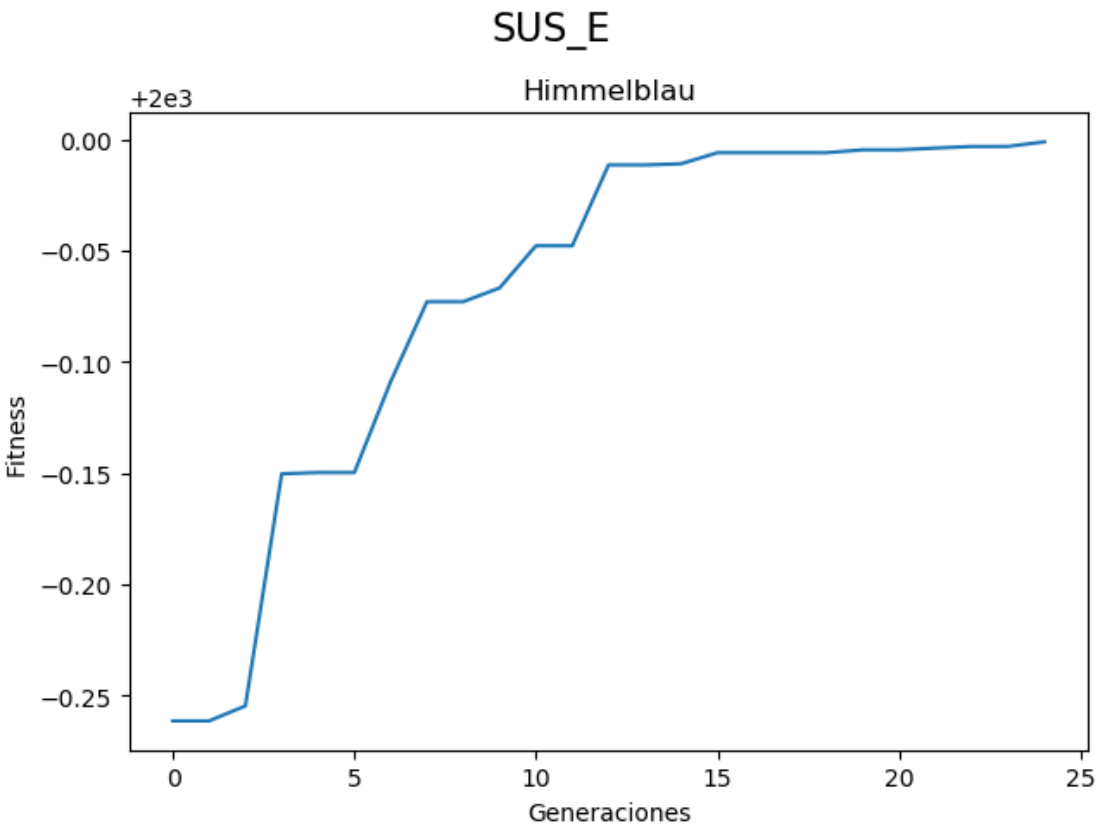
2.7.3. Himmelblau

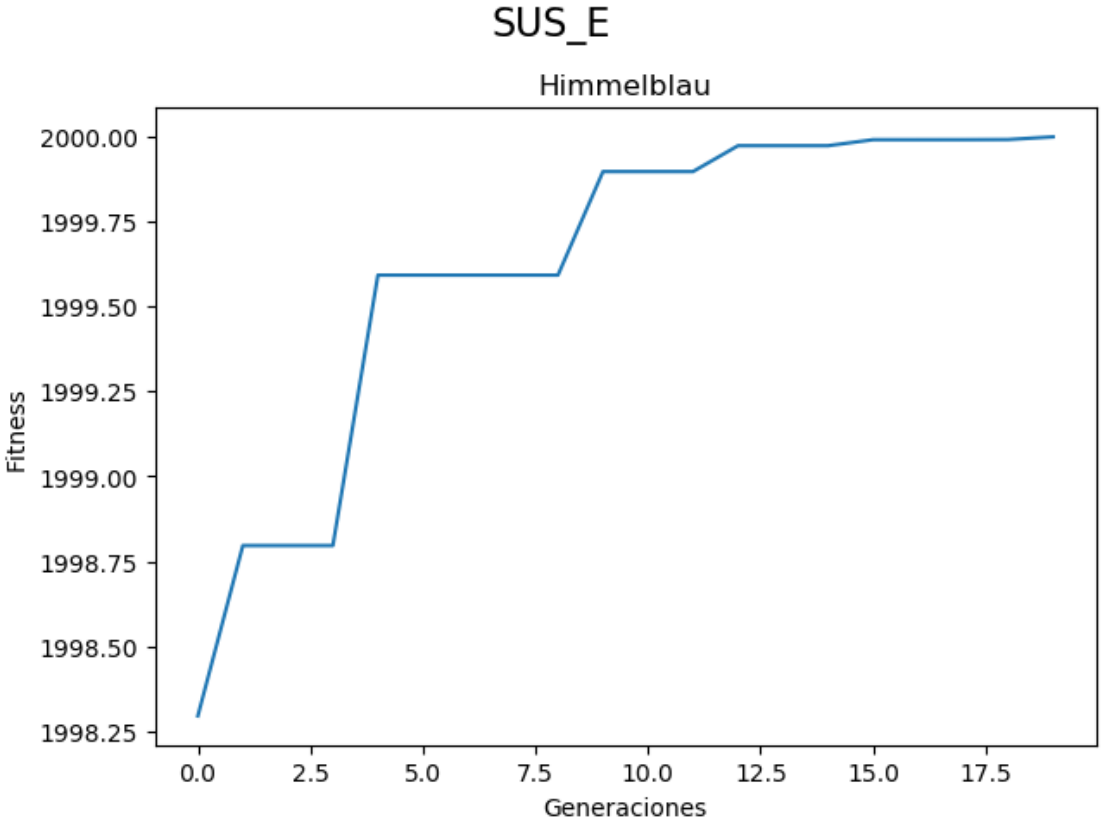
Run: 0
Selection method: SUS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[2.99917602 2.00330355]]
after: 45 generations.
Run: 1
Selection method: SUS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[-3.77799055 -3.27925323] [-3.77799055 -3.27986359]]
after: 3768 generations.
Run: 2
Selection method: SUS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.58511799 -1.85636029]]

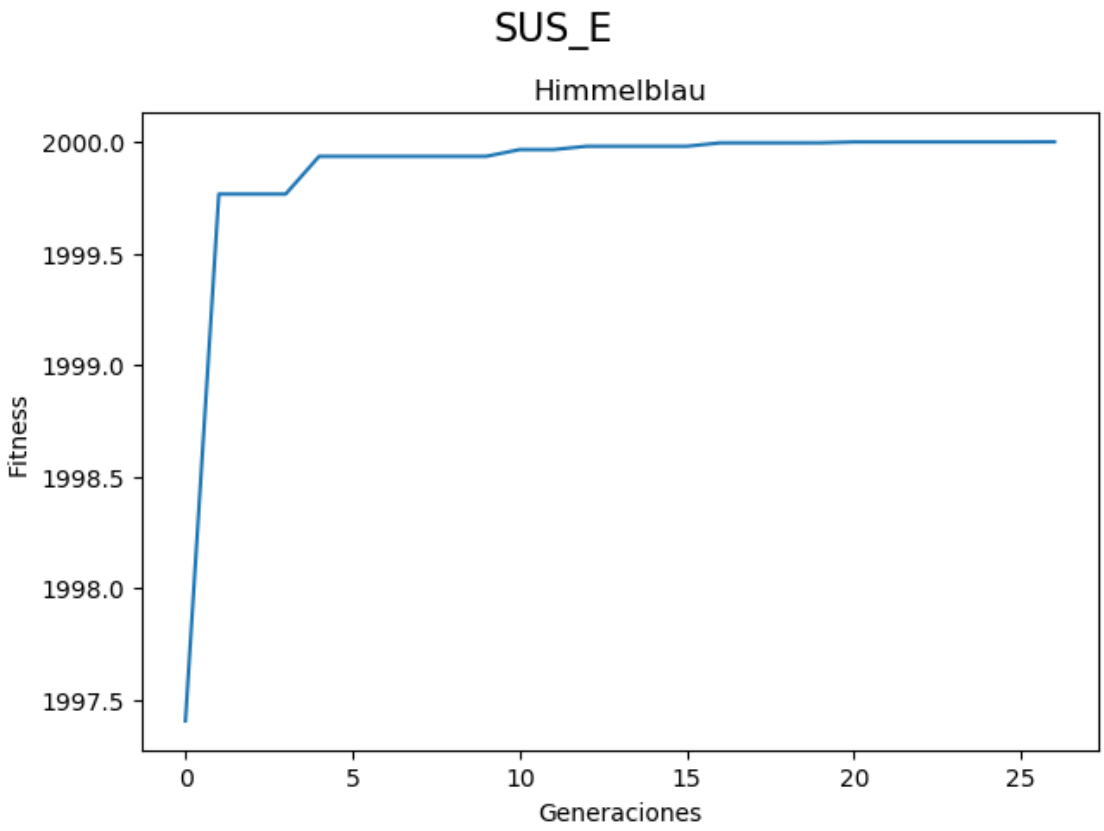
after: 25 generations.
Run: 3
Selection method: SUS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.00062562 2.00627904]]
after: 20 generations.
Run: 4
Selection method: SUS_E
Function to optimize: Himmelblau
Parameters: 'n_individuals': 100, 'pc': 0.95, 'pm': 0.0625, 'max_iter': 50000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[3.00352481 1.99262232]]
after: 27 generations.


SUS_E
Himmelblau
2000
1999
1998
1997
1996
1995
1994
1993
Fitness
0
10
20
30
40
Generaciones


SUS_E
Himmelblau
+2e3
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
Fitness
0
500
1000
1500
2000
2500
3000
3500
Generaciones


SUS_E
Himmelblau
+2e3
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
Fitness
0
5
10
15
20
25
Generaciones

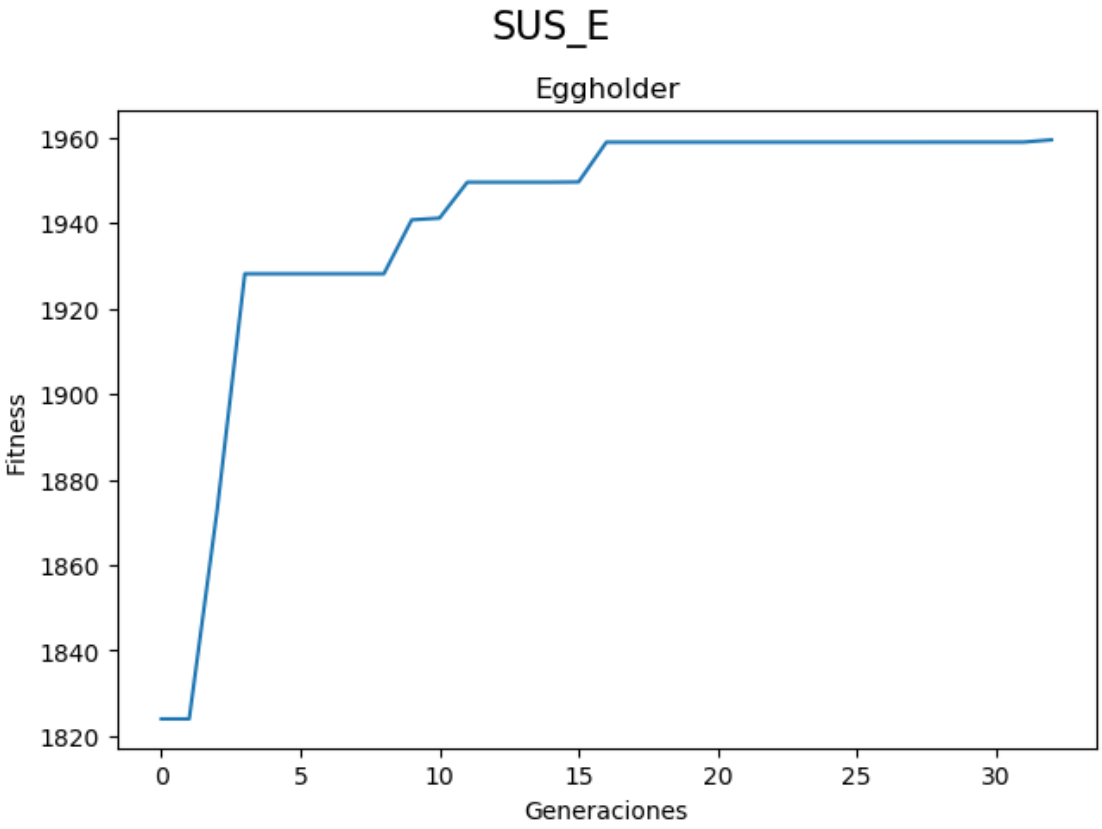

SUS_E
Himmelblau
2000.00
1999.75
1999.50
1999.25
1999.00
1998.75
1998.50
1998.25
Fitness
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Generaciones

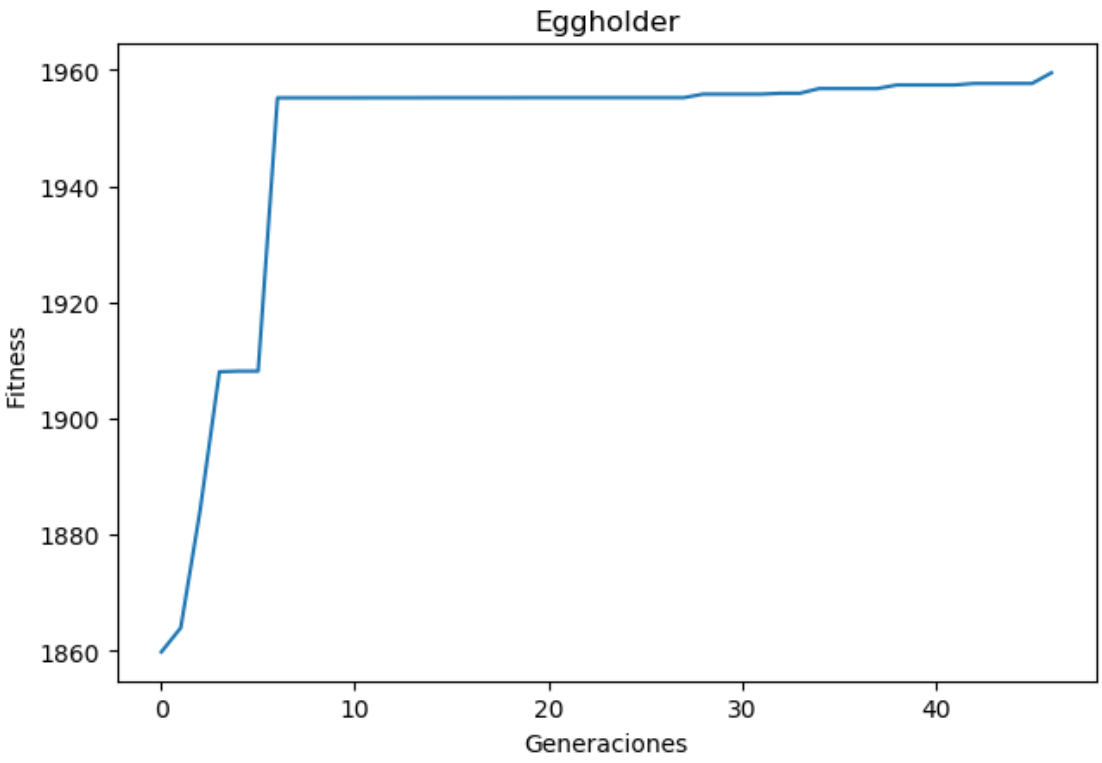


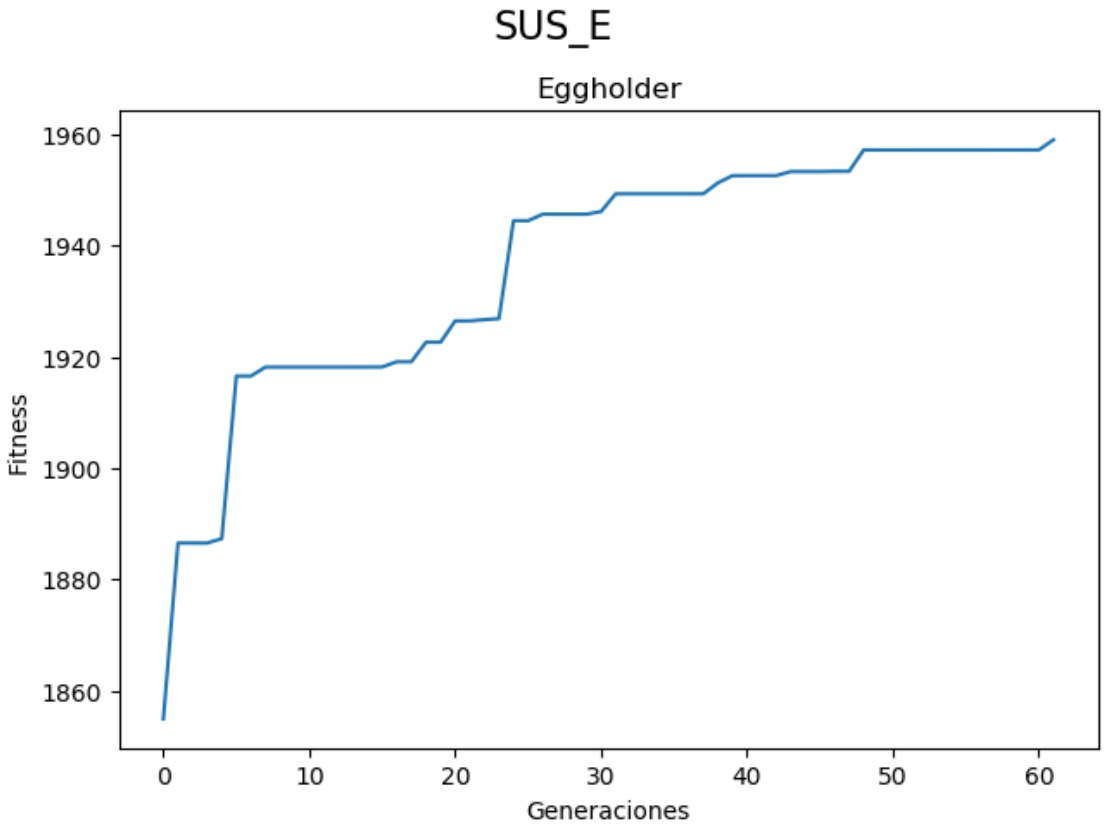
2.7.4. Eggholder

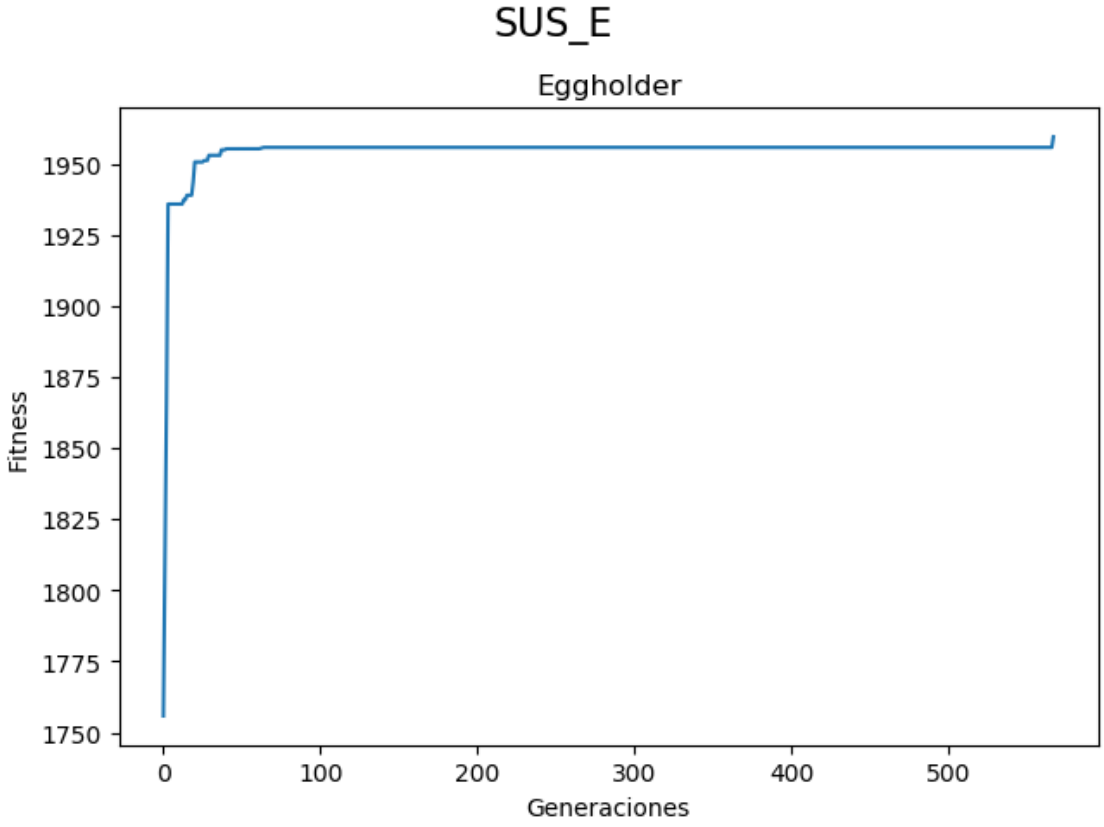
Run: 0
Selection method: SUS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[511.97857666 403.92711758]]
after: 33 generations.
Run: 1
Selection method: SUS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[511.92059326 403.83745693] [511.99432373 403.90410733]]
after: 47 generations.
Run: 2
Selection method: SUS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[511.92736816 403.57830164]]

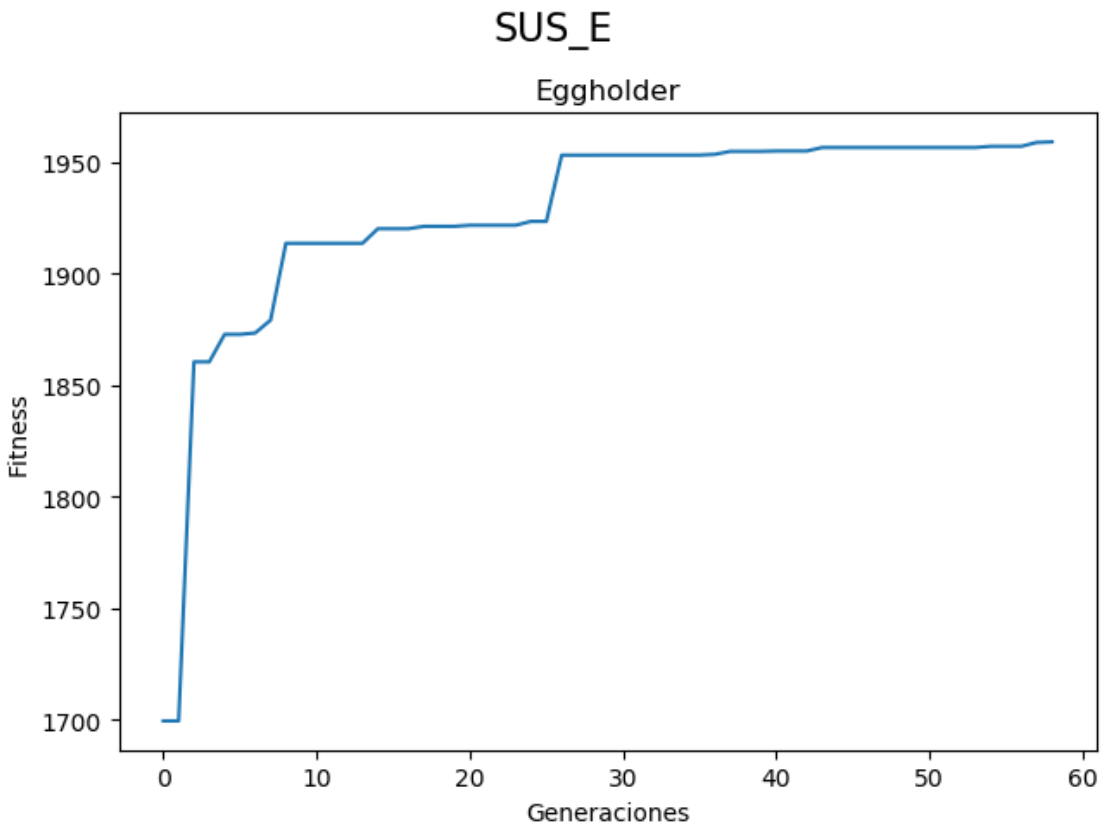
after: 62 generations.
Run: 3
Selection method: SUS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[511.99676514 404.05346036]]
after: 568 generations.
Run: 4
Selection method: SUS_E
Function to optimize: Eggholder
Parameters: 'n_individuals': 500, 'pc': 0.95, 'pm': 0.09090909090909091, 'max_iter': 100000, 'selection': 'sus', 'elitism': 0.2
Global minimum found: [[511.94439697 404.70342378] [511.94476318 403.68578456]]
after: 59 generations.


SUS_E
Eggholder
Fitness
Generaciones
1960
1940
1920
1900
1880
1860
1840
1820
0
5
10
15
20
25
30


SUS_E
Eggholder
Fitness
Generaciones
1960
1940
1920
1900
1880
1860
0
10
20
30
40


SUS_E
Eggholder
1960
1940
1920
1900
1880
1860
Fitness
0
10
20
30
40
50
60
Generaciones


SUS_E
Eggholder
1950
1925
1900
1875
1850
1825
1800
1775
1750
Fitness
0
100
200
300
400
500
Generaciones



3. Discusión.

3.1. Rastrigin.

Se obtuvo el mejor resultado con selección por torneo con elitismo, probablemente porque es una función con un sólo mínimo global, siendo la corrida con menor número de generaciones la primera con 24 y mayor número la segunda con 245.
Con selección proporcional sin elitismo, en general la convergencia fue lenta, requiriendo en 4 de las 5 corridas al menos 9,000 corridas, y en una de ellas ni siguiera se convergió al mínimo global.
La selección proporcional con elitismo tuvo un desempeño similar a la selección por torneo con elitismo, siendo la corrida con menor número de generaciones la tercera con 27 y mayor número la quinta con 42.
La selección por torneo sin elitismo tuvo un desempeño similar a la selección proporcional sin elitismo, siendo la corrida con menor número de generaciones la primera con 3812, y de nuevo, una de las corridas, la tercera no convergió al mínimo global.

La selección por SUS sin elitismo tuvo un desempeño similar a la selección proporcional sin elitismo, siendo la corrida con menor número de generaciones la primera con 300, y de nuevo, una de las corridas, la cuarta no convergió al mínimo global.

La selección por SUS con elitismo tuvo un desempeño similar a la selección por torneo con elitismo, siendo la corrida con menor número de generaciones la cuarta con 27 y mayor número la tercera con 25.
.

3.2. Beale.

Para Beale, de nuevo se obtuvo el mejor resultado aplicando elitismo, pero esta vez con selección proporcional, sin embargo, pese a que los resultados no son tan buenos sin elitismo, todas las corridas sin elitismo con todas las variantes de selección convergieron al mínimo global.

3.3. Himmelblau.

Para Himmelblau, se obtuvieron resultados similares que para Beale, de nuevo se obtuvo el mejor resultado aplicando elitismo, pero esta vez con selección por SUS, y de nuevo pese a que los resultados no son tan buenos sin elitismo, todas las corridas sin elitismo con todas las variantes de selección convergieron al mínimo global.

3.4. Eggholder.

Para eggholder, al ser una función tan accidentada e irregular, manejando una probabilidad más alta de mutación y selección proporcional con elitismo, pareciera encontrarse un mejor resultado. De nuevo pareciera que la mayor velocidad de convergencia se alcanza aplicando elitismo, y de nuevo, las corridas sin elitismo con todas las variantes de selección convergieron al mínimo global, además pareciera que, relativo a las variantes sin elitismo de las otras funciones, se da una convergencia más rápida al optimizar sin elitismo para esta función.