TRABAJO PRÁCTICO N°1 - ALGORITMO GENÉTICO CANÓNICO

CICLO LECTIVO 2025

Mercé Alexis
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 50174
alexis.am.2001@gmail.com

Neirotti Bruno
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 47792
bneirotti@gmail.com

Sebben Mateo
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49609
mateosebben114@gmail.com

5 de junio de 2025

RESUMEN

El presente trabajo se analiza el desempeño de un Algoritmo Genético Canónico aplicado a la búsqueda del valor máximo de una función f(x). Se evalúan diferentes métodos de selección: Ruleta, Torneo y Elitismo, y se estudia el impacto de parámetros como la probabilidad de cruce, mutación, tamaño de población y número de generaciones. Se realizan múltiples ejecuciones del algoritmo (20, 100 y 200 corridas), generando estadísticas y gráficas de máximos, mínimos y promedios por generación. Finalmente, se comparan los resultados obtenidos para cada método de selección y configuración, con el fin de analizar su influencia en la convergencia del algoritmo y la calidad de las soluciones.

Índice

1. Introducción

El presente trabajo práctico tiene como objetivo implementar, analizar y evaluar el comportamiento de un Algoritmo Genético Canónico aplicado a la búsqueda del valor máximo de una función f(x). A lo largo del desarrollo, se abordan tanto los aspectos teóricos como los prácticos de los algoritmos evolutivos, haciendo especial foco en los mecanismos de selección —Ruleta, Torneo y Elitismo— y su influencia sobre la convergencia y la calidad de las soluciones obtenidas.

Se detalla la metodología de desarrollo adoptada, las herramientas de programación utilizadas y la forma de trabajo adoptada. Asimismo, el trabajo incluye el código fuente debidamente comentado, la exposición de resultados obtenidos en múltiples ejecuciones del algoritmo, y el análisis gráfico de la evolución de los valores de fitness a lo largo de generaciones. Finalmente, se presentan conclusiones fundamentadas a partir del estudio comparativo de los métodos de selección y la variación de parámetros como la probabilidad de cruce y de mutación.

1.1. Enunciado del Trabajo

Hacer un programa que utilice un Algoritmo Genético Canónico para buscar un máximo de la función:

$f(x) = \left(\frac{x}{\text{coef}}\right)^2$ en el dominio $[0, 2^{30} - 1]$

donde coef $= 2^{30} - 1$

teniendo en cuenta los siguientes datos:

- Probabilidad de Crossover = 0,75
- Probabilidad de Mutación = 0,05
- Población Inicial: 10 individuos
- Ciclos del programa: 20
- Método de Selección: Ruleta
- Método de Crossover: 1 Punto
- Método de Mutación: invertida

Opción A: El programa debe mostrar, finalmente, el Cromosoma correspondiente al valor máximo, el valor máximo, mínimo y promedio obtenido de cada población. Mostrar la impresión de las tablas de mínimos, promedios y máximos para 20, 100 y 200 corridas. Deben presentarse las gráficas de los valores Máximos, Mínimos y Promedios de la función objetivo por cada generación luego de correr el algoritmo genético 20, 100 y 200 iteraciones (una gráfica por cada conjunto de iteraciones). Realizar comparaciones de las salidas corriendo el mismo programa en distintos ciclos de corridas y además realizar todos los cambios que considere oportunos en los parámetros de entrada de manera de enriquecer sus conclusiones.

Opción B: aplicar lo enunciado en la opción A pero con método de Selección de Torneo.

Opción C: Se entiende por elite a un grupo pequeño que por algún motivo, característica, facultad o privilegio es superior o mejor en comparación al grueso de una población determinada; con cualidades o prerrogativas de las que la gran mayoría no disfrutan.

Un algoritmo genético, desde el punto de vista de la optimización, es un método poblacional de búsqueda dirigida basada en probabilidad. Bajo una condición bastante débil, que el algoritmo mantenga elitismo, es decir, guarde siempre al mejor elemento de la población sin hacerle ningún cambio, se puede demostrar que el algoritmo converge en probabilidad al óptimo. En otras palabras, al aumentar el número de iteraciones, la probabilidad de tener el óptimo en la población tiende a uno.

Luego el método más utilizado para mejorar la convergencia de los algoritmos genéticos es el elitismo.

Este método consiste básicamente para nuestro trabajo en realizar la etapa de selección de la siguiente manera:

* Se realiza un muestreo en una élite de “ere” miembros es decir para nuestro ejercicio se seleccionan dos cromosomas que posean el mejor fitness de entre los mejores de la población inicial y se incorporan directamente a la población siguiente, sin pasar por la población intermedia.

*El proceso se repite para cada población que se va generando hasta completar el número de veces que se ejecutará el algoritmo genético. Se solicita la ejecución de 100 iteraciones.

1.2. Conceptos teóricos

Óptimo local: Es una solución que supera a todas las soluciones vecinas, pero no necesariamente es la mejor dentro de todo el espacio de búsqueda. En otras palabras, es cuando la función objetivo alcanza un valor máximo o mínimo - según se busque - pero existen otros candidatos con un valor aún mejor.

Óptimo global: Es la mejor solución posible en todo el espacio de búsqueda; ningún otro punto tiene un valor superador.

Explotación: Es el proceso de refinar y mejorar soluciones ya conocidas, concentrándose en áreas del espacio de búsqueda que ya mostraron buenos resultados.

Exploración: Es la búsqueda de nuevas regiones del espacio de soluciones, con el objetivo de mantener diversidad y evitar quedar atrapado en óptimos locales.

Espacio de soluciones: Son todos los posibles individuos que pueden generarse dadas las codificaciones y restricciones del problema. Cada individuo de la población representa un punto dentro de este espacio.

2. Descripción de la metodologías de desarrollo abordadas

El algoritmo genético desarrollado adopta un esquema generacional clásico, en el cual una población inicial de soluciones candidatas (cromosomas) evoluciona a lo largo de múltiples generaciones. En cada iteración, se aplican operadores genéticos con el objetivo de mejorar la calidad global de la población, atendiendo una función objetivo que evalúa el desempeño o aptitud de cada individuo. Para llevar a cabo esta evolución, se define un conjunto estructurado de componentes clave: una codificación binaria de los individuos, una función de evaluación adaptada al problema, y un conjunto de operadores de selección, cruce y mutación.

Este trabajo se ha enfocado especialmente en analizar y comparar tres mecanismos de selección de padres, los cuales constituyen el núcleo del proceso evolutivo y determinan en gran medida la dinámica de convergencia del algoritmo. Los tres mecanismos de selección abordados en esta implementación son: Selección de Ruleta, Selección de Torneo, y Elitismo (aplicado a Selección de Ruleta).

2.1. Selección de Ruleta:

Esta técnica modela una ruleta en la cual cada individuo ocupa un sector cuyo tamaño es proporcional a su fitness: cuanto más apto es, más chances tiene de ser seleccionado. Esta técnica favorece a los individuos más aptos pero no excluye completamente a los de menor calidad, permitiendo así un equilibrio entre exploración (variedad) y explotación (refinamiento).

La principal limitación de este método es la posibilidad de convergencia prematura: si uno o pcoos individuos presentan un fitness significativamente superior al resto, pueden dominar rápidamente la población. Esto puede ser perjudicial si esos individuos representan óptimos locales.

2.2. Selección de Torneo:

Este método consiste en seleccionar aleatoriamente un conjunto reducido de individuos (en este caso, un 40 %, lo que corresponde a 4 individuos) y elegir entre ellos al que presente el mayor valor de aptitud, repitiendo este proceso hasta alcanzar la cantidad de padres requerida.

A diferencia de la ruleta, el torneo no se basa en probabilidades proporcionales al fitness absoluto, sino que realiza comparaciones locales. Además, permite ajustar la presión selectiva modificando el tamaño del torneo: torneos más grandes privilegian aún más a los individuos de mayor fitness. Sin embargo, si el tamaño del torneo es demasiado grande se producirá una convergencia prematura hacia óptimos locales. Esta metodología resulta útil para preservar diversidad genética sin perder capacidad de exploración.

2.3. Selección de Elitismo aplicado a Ruleta:

Esta estrategia consiste en preservar los mejores individuos sin alteración alguna, copiándolos directamente a la nueva población (en este caso, un 20 % de la población, lo que corresponde a 2 individuos). Esta metodología asegura que la mejor solución hallada hasta el momento no se pierda debido al aleatoriedad inherente de las operaciones de cruce o mutación.

Desde una perspectiva teórica, el elitismo favorece directamente a las mejores soluciones encontradas en cada generación. En otras palabras, esta técnica genera una presión selectiva positiva, ya que siempre se mantienen los individuos de mayor calidad, permitiendo que el algoritmo avance más rápido hacia las mejores soluciones (óptimo global).

No obstante, esta ventaja también puede derivar en una pérdida de diversidad genética si el número de individuos elitistas es elevado o si el algoritmo no incluye mecanismos para diversificar la población.

3. Descripción de las herramientas de programación utilizadas

3.1. Lenguaje de Programación

El programa está desarrollado utilizando Python, un lenguaje de programación de alto nivel, interpretado y de propósito general. La elección de Python se debe a su sintaxis clara y legible, lo que agiliza el desarrollo y mantenimiento del código, especialmente en la implementación de algoritmos complejos como los genéticos. Además, su amplia variedad de librerías lo convierte en una opción robusta para tareas que involucran cálculos numéricos, manipulación de datos y visualización.

3.2. Librerías utilizadas

Para el desarrollo de este trabajo práctico se han empleado las siguientes librerías de Python:

- **random**: Esta librería estándar de Python es fundamental para la operación de cualquier algoritmo genético, ya que introduce el componente estocástico necesario para la exploración del espacio de soluciones. Se utiliza específicamente para:
  - La inicialización aleatoria de los individuos en la población (generando cadenas de bits al azar).
  - La selección de puntos de cruce (crossover) y la determinación de la probabilidad de cruce.
  - La aplicación de mutaciones en puntos aleatorios del cromosoma de un individuo.
  - La simulación del giro de la ruleta en el método de selección por ruleta.
  - La selección de individuos para el “ring” en el método de selección por torneo.

- **matplotlib.pyplot**: Conocida como pyplot dentro del paquete matplotlib, esta es una librería de visualización de datos ampliamente utilizada en Python. Su aplicación en este programa es crucial para graficar la evolución del algoritmo genético a lo largo de las generaciones. Además permite:
  - Crear y configurar los gráficos, incluyendo títulos, etiquetas de ejes y rangos.
  - Personalizar la apariencia de los ejes, como la frecuencia de las marcas (ticks) en el eje X, para una mejor legibilidad de gráficos con diferentes números de generaciones.
  - Añadir una leyenda para identificar claramente cada línea en el gráfico.
  - Mostrar una cuadrícula en el fondo del gráfico para facilitar la lectura de los valores.

- **mplcursors**: Esta es una librería que se utiliza para mejorar la experiencia del usuario al visualizar los gráficos, permitiendo mostrar información detallada (como los valores exactos) de los puntos trazados en el gráfico cuando el usuario pasa el ratón sobre ellos. Esto es particularmente útil para analizar los valores de fitness en generaciones específicas sin necesidad de consultar tablas de datos.

4. Descripción de la forma de trabajo abordada en equipo

Dado el contexto de virtualidad en el que se desarrolló el trabajo práctico, se optó por una organización colaborativa y estructurada del equipo mediante diferentes herramientas digitales. La comunicación diaria se sostuvo mayormente vía WhatsApp para una coordinación ágil, y realizamos llamadas mediante Discord para compartir archivos y trabajar en el código. Asimismo, abordamos el informe mediante la plataforma Overleaf en formato LaTeX.

El código se desarrolló utilizando el editor de código Visual Studio Code, lo que facilitó la colaboración simultánea en el código fuente, el cual fue implementado en lenguaje Python, como fue mencionado anteriormente. Principalmente la forma de trabajo fue conjunta, lo que si bien inicialmente pudo resultar algo lento, rápidamente luego nos permitió avanzar en el desarrollo del código, ya que al estar trabajando todos al mismo tiempo en comunicación se pudieron resolver dudas y errores con mayor facilidad, y aprender más rápido a partir de ellos.

5. Código del programa

```python
import random
import matplotlib.pyplot as plt
import mplcursors

COEF = 2**30 - 1
LONG_CROM = 30
LONG_POBLACION = 10
PC = 0.75
PM = 0.05
```

Importación de librerías (explicadas anteriormente) y definición de constantes del programa. La variable 'COEF' se utiliza al momento de evaluar la función objetivo en estudio, la variable 'LONG_CROM' determina la longitud del cromosoma (individuo), la variable 'LONG_POBLACION' establece la cantidad de individuos en nuestra población, y por último las variables 'PC' y 'PM' establecen la probabilidad de crossover y mutación respectivamente. Utilizar estas variables globales permite reutilizar este programa haciendo cambios mínimos.

```python
# Menu principal
def menu_ppal():
    global GENERACIONES
    global SELECCION
    while True:
        print("\n--- Cantidad de Generaciones ---")
        print("1. 20 generaciones")
        print("2. 100 generaciones")
        print("3. 200 generaciones")

        opcion = input("Elegi una opcion (1, 2 o 3): ")

        if opcion == "1":
            GENERACIONES = 20
            break
        elif opcion == "2":
            GENERACIONES = 100
            break
        elif opcion == "3":
            GENERACIONES = 200
            break
        else:
            print("Opcion invalida. Proba de nuevo.\n")

    while True:
        print("\n--- Tipo de selecciones disponibles ---")
        print("1. Ruleta")
        print("2. Torneo")
        print("3. Elitismo (aplicado a ruleta)")

        opcion = input("Elegi una opcion (1, 2 o 3): ")

        if opcion == "1":
            SELECCION = 'R'
            break
        elif opcion == "2":
            SELECCION = 'T'
            break
        elif opcion == "3":
            SELECCION = 'E'
            break
        else:
            print("Opcion invalida. Proba de nuevo.\n")
```

Definimos la función del menú principal, que se llama al comienzo de la ejecución del programa para especificar la cantidad de generaciones y el tipo de selección deseada. Se definen las variables globales 'GENERACIONES', que podrá asumir 20, 100 o 200 como valor y que se utilizará más adelante para iterar nuestro programa principal; y 'SELECCION', que podrá asumir R, T o E como valor y que se utilizará al momento de abordar un método de selección en el programa principal. Luego, se utilizan dos estructuras de bucle que se repiten hasta que se ingrese una opción válida de las mencionadas anteriormente.

```
# Generacion de individuo
def individuo():
    return [1 if random.random() > 0.5 else 0 for _ in range(LONG_CROM)]    # Genera una lista de
    30 bits con 1s y 0s al azar
```

Definimos la función individuo (cromosoma), que genera una lista de 30 bits (genes), con unos y ceros al azar, utilizando la librería random. La probabilidad de obtener un 1 o un 0 en cada gen del cromosoma es la misma (50 %), donde en caso de que el número aleatorio obtenido sea mayor a 0,5 se utiliza un 1, o en su defecto un 0.

```
# Generacion de poblacion
def pob_ini():
    return [individuo() for _ in range(LONG_POBLACION)]
```

Definimos la función pob_ini, que sirve para generar la población total, llamando a la función inidividuo tantas veces como el tamaño de la población.

```
# Evaluacion y conversion
def pasaje(ind):
    cadena = ''.join(str(bit) for bit in ind)   # Convierte una lista de bits en una cadena unica
    tipo string
    valor = int(cadena, 2)  # Convierte la cadena binaria a numero entero, sabiendo que estan en
    base 2 (binario)
    return valor
```

Definimos la función pasaje, que recibe un individuo expresado en binario como parámetro, y que devuelve dicho número en decimal. Para ello, primero convierte el parámetro recibido de una lista de bits a una única cadena (string), y luego hace uso de la función nativa int, que recibe una cadena y una base (2, es decir, binario) y la convierte a numero entero decimal.

```
def func_obj(x):
    return (x / COEF) ** 2
```

Esta función lo que permite es calcular el valor del individuo pasandolo por parámetro ("x"), se calcula el valor de este en la función objetivo planteada en el enunciado.

```
def calculaFitness(poblacion):
    # Calcula el fitness relativo de cada individuo
    decimales = [pasaje(ind) for ind in poblacion]  # Para cada individuo de la poblacion hacemos
    el pasaje y lo guardamos en el array decimales
    objetivos = [func_obj(x) for x in decimales]    # Para cada individuo de la poblacion (ya en
    decimal) calculamos su funcion objetivo y lo guardamos en el array objetivos
    suma_total = sum(objetivos)
    if suma_total == 0:
        # Si todo es cero (muy raro, pero hay que contemplarlo ya que la division por cero no esta
    definida), repartimos el fitness de forma pareja
        fitness = [1 / LONG_POBLACION for _ in objetivos]
    else:
        # Si no, cada uno recibe su parte proporcional (peso que tiene cada uno dentro de la
    poblacion)
        fitness = [obj / suma_total for obj in objetivos]
    return fitness
```

Definimos la función calculaFitness, que recibe como parámetro a la población entera, y que devuelve un arreglo con el fitness (peso relativo de cada individuo) correspondiente. Primeramente, pasamos la población de binario a decimal usando la función pasaje, luego evaluamos cada indidivuo de dicha población según la función objetivo y finalmente, teniendo en cuenta la sumatoria total de las funciones objetivos obtenidas, se calcula el peso relativo de cada individuo de la población (fitness).

```
# Seleccion por ruleta
def seleccionRuleta(poblacion, fitness):
    """
    Se basa en el peso relativo de cada individuo como probabilidad de salir elegido (puede ser elegido incluso mas de una vez)
    """
    acumulado = []  # Aca vamos a ir guardando la suma acumulada de los fitness
    suma = 0
    for f in fitness:
        suma += f
        acumulado.append(suma)  # Ya tenemos el array acumulado completado

    seleccion = []  # Aca vamos a guardar los padres que generaran la siguiente generacion
    for _ in range(LONG_POBLACION): # Vamos a seleccionar tantos individuos como el tamano de la poblacion
        r = random.random() # Numero aleatorio entre 0 y 1
        for (idx, valorAcum) in enumerate(acumulado):    # Enumerate() devuelve tanto el id como el valor en cada posicion que recorre del arr
            if r <= valorAcum:  # Si el numero aleatorio generado es menor o igual al valor acumulado entonces
                seleccion.append(poblacion[idx].copy())   # Copiamos al individuo seleccionado en la posicion coincidente. Usamos copy() para crear un clon del individuo, no una referencia al array original
                break
    return seleccion
    """
    Ejemplo:
    Array fitness: [0.15, 0.12, 0.10, 0.08, 0.05, 0.20, 0.07, 0.03, 0.10, 0.10]
    Array acumulado: [0.15, 0.27, 0.37, 0.45, 0.50, 0.70, 0.77, 0.80, 0.90, 1.00]
    Numero aleatorio: 0.49
    Enumerate(acumulado): [(0, 0.15),(1, 0.27),(2, 0.37),(3, 0.45),(4, 0.50),(5, 0.70),...,(9, 1.00)]
    0.49 <= 0.15? ---> NO
    ...
    0.49 <= 0.5 ---> Si: selecionar ese individuo en la id donde estaba 0.50 (4). Ese individuo sera un padre. Romper el ciclo FOR interno.
    Este proceso de seleccion se cumplio bien ya que el individuo seleccionado tenia 5% de chances, y el acumulado que posibilitaba estaba un numero entre 0.45 y 0.5 (0.05)
    """
```

Definimos la funcion de seleccionRuleta, que recibe como parámetro la población (en binario) y el fitness de dicha población, y devuelve un nuevo arreglo (del mismo tamaño que la población) compuesto por quienes serán los padres de la próxima generación (donde se permite que un mismo individuo esté repetido).

En primer lugar, se utiliza un array acumulado, donde iremos guardando la suma acumulada a medida que recorremos los valores del arreglo fitness. De esta manera, el último valor del arreglo será 1.

A continuación se itera tantas veces como el tamaño de la población, se obtiene un número aleatorio entre cero y uno y se realiza otra iteración teniendo en cuenta tanto la posición (idx) como el valor acumulado (valorAcum) hallado para el arreglo. Se puede obtener tanto la posición como el propio valor de cada elemento de un arreglo utilizando la función nativa enumerate.

Luego, se compara el número aleatorio obtenido contra el valorAcum. En caso que el número sea menor o igual a dicho valor, entonces el individuo en la posicion coincidente (idx) deberá formar parte del arreglo de padres (seleccion) rompiendo la iteración en curso. De lo contrario, se sigue avanzando en la iteración y evaluando la condición hasta que se cumpla. Finalmente, se devuelve el arreglo de padres completo.

```python
def seleccionTorneo(poblacion):
    """
    Elige al mejor entre el 40% de la poblacion al azar, y esto se repite tantas veces como el tamano de la poblacion.

    Algo a tener en cuenta es que si incluimos prints en los individuos que van subiendo al ring podriamos ver que se suba dos veces el mismo individuo.
    """
    seleccion = []
    cantidadGrupo = int(len(poblacion) * 0.4)

    for _ in range(LONG_POBLACION):  # Hace un for iterando tantas veces como la longitud de la poblacion
        ring = []

        while len(ring) < cantidadGrupo: # Bucle que se asegura que sean siempre 40% de individuos dentro de "ring"
            idx = random.randint(0, LONG_POBLACION - 1)
            ring.append(poblacion[idx].copy())

        fitnessRing = calculaFitness(ring) # Calcula el fitness de cada uno de los individuos dentro del "ring"
        mejor_idx = fitnessRing.index(max(fitnessRing)) # Encuentra cual es el indice del individuo que tiene el mayor fitness en el ring.
        seleccion.append(ring[mejor_idx].copy()) # Toma al individuo ganador(el que tiene mejor fitness) y lo agrega a la lista "seleccion"
    return seleccion
```

Definimos la funcion de seleccionTorneo, que recibe como parámetro la población (en binario) y devuelve un nuevo arreglo (del mismo tamaño que la población) compuesto por quienes serán los padres de la próxima generación (donde se permite que un mismo individuo esté repetido).

Inicialmente, se establece que el tamaño del grupo que realizará cada torneo sea el 40 % del total de la población (para el caso en estudio, 4 individuos). Luego, se itera tantas veces como el tamaño de la población y se define un arreglo ring, que se irá completando con individuos aleatorios de la población hasta alcanzar el tamaño deseado, incluso pudiendo repetirse un mismo individuo. A continuación, se calcula el fitness de cada individuo del ring obtenido, y sabiendo eso se determina la posición del individuo con mayor fitness (utilizando las funciones nativas del lenguaje max e index), para luego agregarlo al arreglo de padres (seleccion).

```python
def seleccionElitismo(poblacion, fitness, cantidad_elite):
    """
    El 20% mejor de la poblacion pasara directamente a la siguiente generacion, entre el resto aplicamos el mtodo de Ruleta para su seleccion
    """

    # Seleccionamos los 20% mejores individuos de la poblacion
    fitness_y_poblacion = list(zip(fitness, poblacion))   # Vincular cada fitness calculado a su correspondiente individuo, y lo hacemos lista para poder manejarlo como un array de elementos (x, y)
    fitness_y_poblacion.sort(reverse=True)  # ordena por el primer elemento de cada tupla automaticamente (fitness)
    pob = [individuo for _, individuo in fitness_y_poblacion]  # Ahora volvemos a obtener el array de poblacion, pero ya ordenado de mayor a menor fitness
    fitness_pob = [fit for fit, _ in fitness_y_poblacion]
    elite = [ind.copy() for ind in pob[:cantidad_elite]]    # Agarramos solo los dos mejores

    # Calcular fitness acumulado (para ruleta)
    acumulado = []
    suma = 0
    for f in fitness_pob:
        suma += f
        acumulado.append(suma)  # Ya tenemos el array acumulado completado

```

```
    # Seleccion por ruleta para completar el resto
    seleccion = elite.copy()
    while len(seleccion) < LONG_POBLACION:
        r = random.random() # Numero aleatorio entre 0 y 1
        for (idx, valorAcum) in enumerate(acumulado):
            if r <= valorAcum:
                seleccion.append(pob[idx].copy())
                break
    return seleccion
```

Definimos la funcion de seleccionElitismo, que recibe como parámetro la población (en binario), el fitness de dicha población y la cantidad de individuos que se tomarán como los mejores y pasan directamente a la próxima generación; y devuelve un nuevo arreglo (del mismo tamaño que la población) compuesto por quienes serán los padres de la próxima generación (donde se permite que un mismo individuo esté repetido).

Primeramente, mediante las funciones nativas zip y list se vincula cada fitness calculado a su correspondiente individuo, y se toma eso como una lista para poder manejarlo correctamente. Luego, se ordena por el primer elemento de cada tupla automaticamente (es decir, por el fitness) utilizando la función nativa sort en reversa (de mayor a menor). Seguidamente, obtenemos la misma población pero en orden de mayor a menor según el fitness, y a su vez obtenemos el fitness asociado a ese nuevo orden de población. Por último, definimos una variable elite, que contendrá los dos mejores individuos de la población ordenada (ya que se había definido que sea el 20 % del total).

Partiendo de esa base, después aplicamos la lógica de selección de ruleta para completar el arreglo de padres. Entonces, se utiliza un array acumulado donde iremos guardando la suma acumulada a medida que recorremos los valores del arreglo fitness. De esta manera, el último valor del arreglo será 1. Se itera hasta completar el arreglo seleccion, se obtiene un número aleatorio entre cero y uno y se realiza otra iteración teniendo en cuenta tanto la posición (idx) como el valor acumulado (valorAcum) hallado para el arreglo. Nuevamente, se puede obtener tanto la posición como el propio valor de cada elemento de un arreglo utilizando la función nativa enumerate.

Luego, se compara el número aleatorio obtenido contra el valorAcum. En caso que el número sea menor o igual a dicho valor, entonces el individuo en la posicion coincidente (idx) deberá formar parte del arreglo de padres (seleccion) rompiendo la iteración en curso. De lo contrario, se sigue avanzando en la iteración y evaluando la condición hasta que se cumpla. Finalmente, se devuelve el arreglo de padres completo.

```
# Cruce
def crossover(poblacionSeleccionada, longitud_poblacion):
    prox_gen = []
    for i in range(0, longitud_poblacion, 2):   # Voy agarrando los padres de a pares
        padre1 = poblacionSeleccionada[i]
        padre2 = poblacionSeleccionada[(i + 1)]
        if random.random() <= PC:
            punto = random.randint(1, LONG_CROM - 1)    # Punto aleatorio donde se va a cortar el
    cruce
            hijo1 = padre1[:punto] + padre2[punto:]
            hijo2 = padre2[:punto] + padre1[punto:]
        else:
            hijo1, hijo2 = padre1.copy(), padre2.copy() # Si no cruzamos, los hijos son iguales a
    los padres

        # Anadimos los 2 hijos generados al array prox_gen
        prox_gen.append(hijo1)
        prox_gen.append(hijo2)
    return prox_gen   # Devuelve los hijos
```

Definimos la funcion de crossover, que recibe como parámetro la selección de padres (en binario) y la cantidad de individuos que se cruzarán en esa población, y devuelve un nuevo arreglo de hijos aún sin mutar.

A partir de una estructura de iteración (que contempla la cantidad recibida por parámetro), se van eligiendo padres de a pares y se evalúa, mediante un número aleatorio, si habrá cruce entre ellos (es decir, si se cumple que el número obtenido es menor o igual a la probabilidad de cruce determinada globalmente). En caso que lo haya, se toma una posición aleatoria entre el 1 y el 29 y se generan los dos hijos en base a los dos padres cruzados respetando dicho punto de corte. Si no hay cruce, los hijos serán una copia de los dos padres. Finalmente, una vez generados tantos hijos como

el tamaño de la población, se devuelve el arreglo.

```
# Mutacion
def mutacion(nuevaPoblacion):
    for ind in nuevaPoblacion:
        if random.random() <= PM:
            punto = random.randint(0, LONG_CROM - 1)     # Elegimos una posicion aleatoria entre 0 y la longitud del cromosoma
            ind[punto] = 1 - ind[punto]  # Invierte solo ese bit
```

Definimos la funcion de mutacion, que recibe como parámetro los hijos obtenidos del crossover (en binario) y devuelve un nuevo arreglo de hijos (la próxima generación).

Se define una estructura de iteración que, para cada individuo de la población recibida (hijos), arroja un número aleatorio entre cero y uno y determina si cumple que sea menor o igual a la probabilidad de mutación definida globalmente. En caso que lo haga, entonces genera un número aleatorio entero entre 0 y 29 que será utilizado como la posición del gen a mutar (invertir dicho bit). De lo contrario, no efectúa cambios en el individuo (no muta).

```
def main():
    menu_ppal()

    # Defino variables para despues el grafico final
    historico_mejor_x = []
    historico_mejor_f = []
    historico_peor_f = []
    historico_promedio_f = []

    poblacion = pob_ini()

    for gen in range(GENERACIONES):
        poblacionDecimal = [pasaje(crom) for crom in poblacion]

        mejor_x = max(poblacionDecimal)
        historico_mejor_x.append(mejor_x)
        peor_x = min(poblacionDecimal)
        promedio_x = sum(poblacionDecimal) / len(poblacionDecimal)

        mejor_f = func_obj(mejor_x)
        historico_mejor_f.append(mejor_f)
        peor_f = func_obj(peor_x)
        historico_peor_f.append(peor_f)
        promedio_f = func_obj(promedio_x)
        historico_promedio_f.append(promedio_f)

        print(f"Generacion {gen+1:03d}: -> Mejor x: {mejor_x:<{12}}  /    {bin(mejor_x)[2:]}    /// Peor x: {peor_x:<{12}}  /    ///    Promedio x: {promedio_x:<{12}.2f}")

        fitnessPob = calculaFitness(poblacion) # Aca tengo los pesos relativos de cada individuo
        if gen < GENERACIONES - 1:
            # Ahora obtengo los que seran padres
            if SELECCION == 'R':
                seleccionados = seleccionRuleta(poblacion, fitnessPob)

            elif SELECCION == 'T':
                seleccionados = seleccionTorneo(poblacion)
            else:
                cantidad_elite = int(LONG_POBLACION*0.2)
                seleccionados = seleccionElitismo(poblacion, fitnessPob, cantidad_elite)
            # En caso de Elitismo, los dos mejores (que estan primeros) pasan directo a la nueva generacion sin crossover ni mutacion
            if SELECCION == 'E':
                elites = seleccionados[:cantidad_elite]  # Los primeros 20% son los elites
```

```
                restantes_crossover = seleccionados[cantidad_elite:]  # Los 80% restantes si se cruzan y mutan
                hijos = crossover(restantes_crossover, (LONG_POBLACION - cantidad_elite)) # Solo vamos a cruzar el 80%, ya que el 20% restante paso directamente por elitismo
                mutacion(hijos)    # Muto solo los hijos surgidos del crossover
                poblacion = elites + hijos  # Nueva generacion: elites + hijos ya mutados
            else:
                nuevaPoblacion = crossover(seleccionados, LONG_POBLACION)   # Cruzamos toda la poblacion
                mutacion(nuevaPoblacion)  # Muto los hijos (mi nueva poblacion)
                poblacion = nuevaPoblacion    # Ya tengo mi nueva poblacion (hijos) para la siguiente generacion

    # Para el resumen final (fuera del bucle)
    mejor_x_total = max(historico_mejor_x)

    print(f"{'-'*100}")
    print(f"{'-'*35}  RESULTADO FINAL  {'-'*35}")
    print(f"{'-'*100}")
    print(f"Maximo x logrado: {mejor_x_total} ({bin(mejor_x_total)[2:]}), que evaluado en f da como resultado: {func_obj(mejor_x_total):.6f}")

    # GRAFICAR evolucion de f(x)
    generaciones = list(range(1, GENERACIONES + 1))

    lienzo, eje = plt.subplots(figsize=(12, 10))

    lineamejor = eje.plot(generaciones, historico_mejor_f, label='f(Mejor_x)')[0]
    lineapeor = eje.plot(generaciones, historico_promedio_f, label='f(Promedio_x)')[0]
    lineaprom = eje.plot(generaciones, historico_peor_f, label='f(Peor_x)')[0]
    eje.set_title('Evolucion de f(x) por Generacion')
    eje.set_xlabel('Generacion')
    eje.set_ylabel('Valor de f(x)')

    if GENERACIONES == 20:
        eje.set_xticks(range(0, GENERACIONES + 1, 1))
    elif GENERACIONES == 100:
        eje.set_xticks(range(0, GENERACIONES + 1, 10))
    else:
        eje.set_xticks(range(0, GENERACIONES + 1, 20))

    eje.legend()
    eje.grid(True)

    plt.tight_layout()

    # Esto permite que al pasar el mouse por los puntos del grafico te diga el valor exacto
    mplcursors.cursor([lineamejor, lineapeor, lineaprom], hover=True)
    plt.show()



if __name__ == "__main__":
    main()
```

Definimos la función principal main, que en primer lugar llama a menu_ppal explicada anteriormente. Luego, se inicializan tres listas vacías que cumplirán un rol importante en el seguimiento y análisis del proceso evolutivo de las gráficas: en ellas se almacenarán, para cada generación, los valores de la función objetivo correspondientes al mejor individuo, peor individuo e individuo promedio, respectivamente.

A continuación, se genera la población inicial mediante la función pob_ini, que devuelve una lista de cromosomas codificados en binario. Tras eso comenzamos el ciclo evolutivo, que se ejecuta una cantidad de veces igual al valor de GENERACIONES definido globalmente. En cada iteración, se convierte la población actual de binario a decimal utilizando la función pasaje, se calcula cuál es el mejor individuo (máximo valor), el peor (mínimo) y el promedio. Cada uno de estos valores se evalúa con la función objetivo func_obj, y los resultados se guardan en las listas previamente creadas para poder graficar luego la evolución de f(x) a lo largo de las generaciones. Simultáneamente, se imprime por consola un resumen de la generación actual, que incluye los valores de x (en decimal y binario) para el mejor, peor y promedio individuo.

Posteriormente, se calcula el fitness de cada individuo de la población mediante la función calculaFitness. Este conjunto de valores es utilizado en la etapa de selección, la cual puede adoptar diferentes estrategias dependiendo del parámetro global SELECCION: si se selecciona 'R', se aplica selección por ruleta; si se elige 'T', se realiza un torneo entre individuos; y si se opta por 'E', se implementa el método de elitismo. En este último caso, se preserva directamente un porcentaje fijo de los mejores individuos (el 20 % de la población), que pasan sin modificaciones a la siguiente generación. El resto de los individuos seleccionados se les aplica las funciones de cruce y mutación. Posteriormente, se realiza el cruce y la mutación mediante sus respectivas funciones.

Finalmente, simplemente se cruzan y mutan todos los seleccionados, y estos pasan a conformar la nueva población. En el caso que se utilice elitismo, se forma la nueva generación uniendo los individuos elite con los hijos ya mutados.

Una vez que se completan todas las generaciones, se realiza una evaluación final para determinar cuál es el mejor individuo entre todas las generaciones. Se imprime su valor decimal, su representación binaria y el resultado obtenido al evaluarlo con la función objetivo. Para finalizar, se genera un gráfico de líneas que muestra la evolución de f(x) en cada generación, tanto para el mejor como para el peor y el promedio de los individuos, utilizando las funciones de la librería matplotlib. Se configuran los ejes, etiquetas y escala según la cantidad de generaciones, y se incorpora una herramienta interactiva que permite visualizar los valores exactos al pasar el cursor por los puntos del gráfico.

6. Gráficas y salidas por pantalla de las corridas

6.1. Método de Ruleta

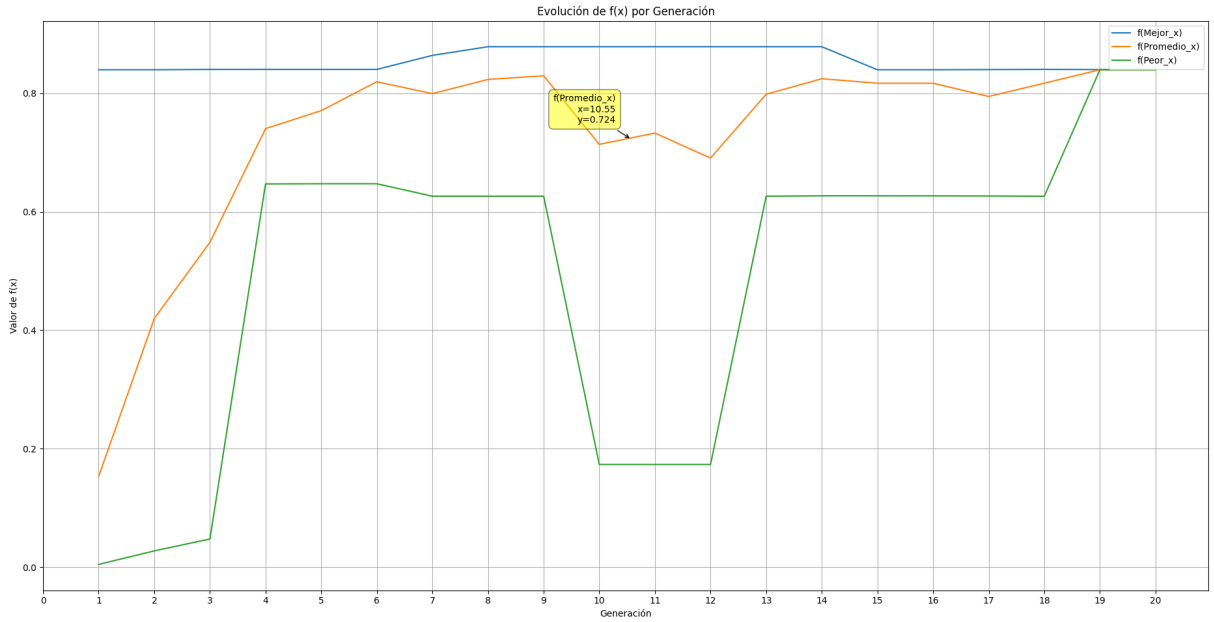

Figura 1: 20 generaciones - Método de Ruleta

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	983878866	11101010100100110011010010	72723118	420109393.8	0.839621449
2	983878866	11101010100100110011010010	177753589	695660499.4	0.839621449
3	984188976	11101010101001100010000110000	233932848	795101754.3	0.840150815
4	984206546	11101010101001110011010010	863602770	923819541.1	0.840180813
5	984206546	11101010101001110011010010	863816881	942574533.4	0.840180813
6	984206546	11101010101001110011010010	863816914	971970917.6	0.840180813
7	998034642	11101101111100110011010010	849660976	959964747.2	0.863955737
8	1006423250	11101111111100110011010010	849660976	974258132.6	0.878540088
9	1006423250	11101111111100110011010010	849660976	977892120	0.878540088
10	1006423248	11101111111100110011010000	446975186	907109963.2	0.878540084
11	1006423250	11101111111100110011010010	446975186	919145665.6	0.878540088
12	1006423250	11101111111100110011010010	446975186	892285704	0.878540088
13	1006423250	11101111111100110011010010	849660978	959358523.2	0.878540088
14	1006423250	11101111111100110011010010	850119888	974998689.8	0.878540088
15	983878866	11101010100100110011010010	850119730	970479998.6	0.839621449
16	983878866	11101010100100110011010010	850087122	970480014.4	0.839621449
17	984042706	11101010100111010011010010	849890514	957104133	0.839901108
18	984239314	11101010101010010011010010	849628368	970506244.8	0.84023676
19	984042706	11101010100111010011010010	983846098	983895250	0.839901108
20	984042706	11101010100111010011010010	983846098	983878866	0.839901108

El mejor valor de x entre todas las generaciones fue 1006423250, y su equivalente en binario es 11101111111100110011010010.

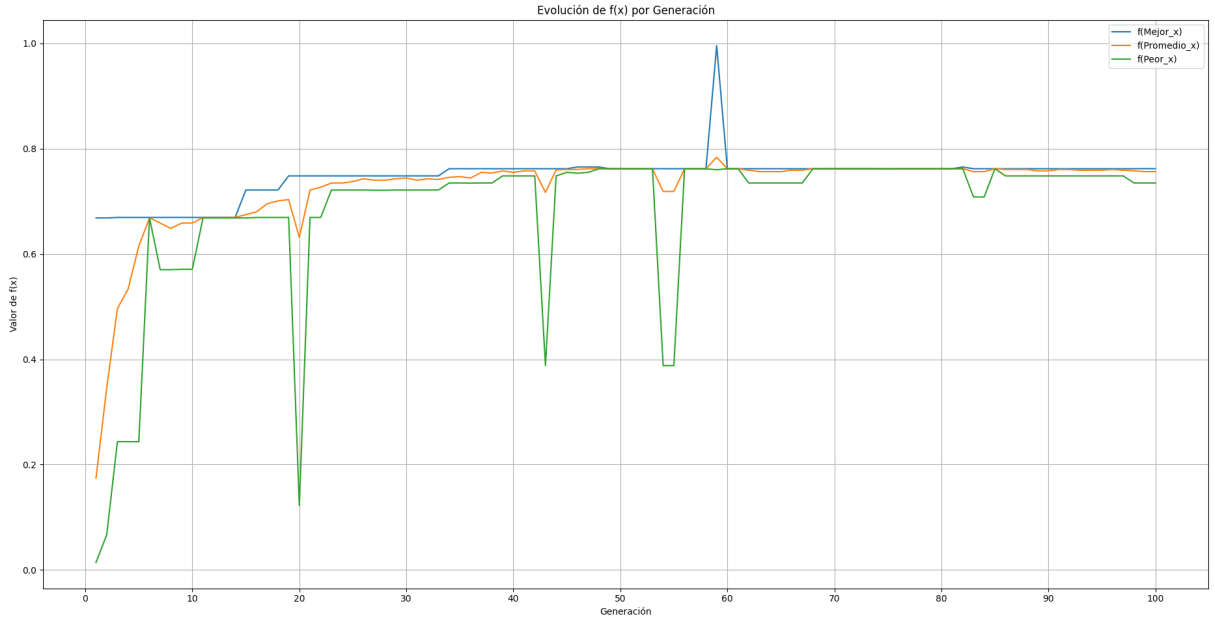

Figura 2: 100 generaciones - Método de Ruleta

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	877977865	11010001010100111000010000 1001	127625421	448562604.6	0.668601574
2	877978534	11010001010100111000111010 0110	277123507	631323328.4	0.668602593
3	878591519	11010001011110001111100001 1111	530018820	756593958	0.669536526
4	878591519	11010001011110001111100001 1111	530018820	784322358.2	0.669536526
5	878591521	11010001011110001111100010 0001	529850633	842203885.7	0.669536529
6	878591521	11010001011110001111100010 0001	877978527	878162359	0.669536529
7	878607905	11010001011110011111100010 0001	810869663	871619201.4	0.6695615
8	878607241	11010001011110011110111000 1001	810869663	864794650.8	0.669560488
9	878607241	11010001011110011110111000 1001	811393311	871566772.2	0.669560488
10	878591519	11010001011110001111100001 1111	811393311	871626501.4	0.669536526
11	878591519	11010001011110001111100001 1111	877978527	878407621.4	0.669536526
12	878591903	11010001011110001111111001 1111	877978143	878346258.2	0.669537111
13	878591519	11010001011110001111100001 1111	877978143	878468626.2	0.669536526
14	878591647	11010001011110001111101001 1111	877978143	878529861.4	0.669536721
15	912145183	11011001011110001110110001 1111	877978143	882101560.6	0.72165263
16	912145951	11011001011110001111100001 1111	878590751	885518264.6	0.721653846
17	912145951	11011001011110001111100001 1111	878590879	895584581.4	0.721653846
18	912145951	11011001011110001111100001 1111	878590623	898933330.2	0.721653846
19	928922399	11011101011110001110110001 1111	878590623	900610927	0.74844369
20	928922399	11011101011110001110110001 1111	375274271	853424901.4	0.74844369
21	928922399	11011101011110001110110001 1111	878590623	912145131.8	0.74844369
22	928922399	11011101011110001110110001 1111	878590623	915500575	0.74844369
23	928922399	11011101011110001110110001 1111	912145055	920533727	0.74844369
24	928922399	11011101011110001110110001 1111	912145055	920533727	0.74844369
25	928922399	11011101011110001110110001 1111	912145055	922211461.4	0.74844369
26	928922399	11011101011110001110110001 1111	912145055	925540677.4	0.74844369
27	928922399	11011101011110001110110001 1111	911883167	923861906.2	0.74844369
28	928922399	11011101011110001110110001 1111	911883167	923861893.4	0.74844369
29	928922399	11011101011110001110110001 1111	912145055	925565816.6	0.74844369
30	928922271	11011101011110001110101001 1111	912145055	926404869.4	0.748443484
31	928922399	11011101011110001110110001 1111	912145055	923888248.6	0.74844369
32	928922399	11011101011110001110110001 1111	912145055	925566034.2	0.74844369
33	928922271	11011101011110001110101001 1111	912145055	924727160.6	0.748443484
34	937310879	11011111011110001110101001 1111	920533535	927242936.6	0.762022117
35	937310879	11011111011110001110101001 1111	920533535	928082603.8	0.762022117
36	937311135	11011111011110001110111001 1111	920402463	926390135.8	0.762022533
37	937311135	11011111011110001110111001 1111	920533535	933114207	0.762022533
38	937302943	11011111011110000110111001 1111	920533535	932272875.8	0.762009213
39	937302943	11011111011110000110111001 1111	928922271	934788690.2	0.762009213
40	937302943	11011111011110000110111001 1111	928922270	933112606.9	0.762009213
41	937311135	11011111011110001110111001 1111	928914079	934788690.2	0.762022533
42	937311135	11011111011110001110111001 1111	928914079	934788715.8	0.762022533
43	937311135	11011111011110001110111001 1111	668867231	909203435.8	0.762022533

44	937311135	11011111011110001110111001111	928914079	936047800.6	0.762022533
45	937311135	11011111011110001110111001111	933108639	936466411.8	0.762022533
46	939408031	11011111111110001110101001111	932059807	936572063	0.765435849
47	939408031	11011111111110001110101001111	933108639	937307704.6	0.765435849
48	939408031	11011111111110001110101001111	937302687	937519007	0.765435849
49	937311135	11011111011110001110111001111	937302687	937309266.2	0.762022533
50	937310879	11011111011110001110101001111	937302687	937310059.8	0.762022117
51	937310879	11011111011110001110101001111	937302687	937309240.6	0.762022117
52	937310879	11011111011110001110101001111	937179807	937294495	0.762022117
53	937310879	11011111011110001110101001111	937179807	937293675.8	0.762022117
54	937319071	11011111011110010110101001111	668867231	910425554.2	0.762035437
55	937319071	11011111011110010110101001111	668867231	910437842.2	0.762035437
56	937319071	11011111011110010110101001111	937171615	937269919	0.762035437
57	937319071	11011111011110010110101001111	937171615	937280568.6	0.762035437
58	937335455	11011111011110100110101001111	937179807	937281387.8	0.762062077
59	1071520415	11111111011110000110101001111	936254111	950626155.8	0.995866585
60	937335455	11011111011110100110101001111	937302559	937319058.2	0.762062077
61	937335455	11011111011110100110101001111	937302687	937315794.2	0.762062077
62	937302687	11011111011110000110101001111	920525471	935624965.4	0.762008797
63	937302687	11011111011110000110101001111	920525471	933947243.8	0.762008797
64	937302687	11011111011110000110101001111	920525471	933947243.8	0.762008797
65	937302687	11011111011110000110101001111	920525471	933948882.2	0.762008797
66	937302687	11011111011110000110101001111	920525471	935624965.4	0.762008797
67	937302687	11011111011110000110101001111	920525471	935624965.4	0.762008797
68	937302687	11011111011110000110101001111	937302687	937302687	0.762008797
69	937368223	11011111011111000110101001111	937302687	937309240.6	0.76211536
70	937368223	11011111011111000110101001111	937300639	937315589.4	0.76211536
71	937368223	11011111011111000110101001111	937302687	937309240.6	0.76211536
72	937368223	11011111011111000110101001111	937302671	937309239	0.76211536
73	937368223	11011111011111000110101001111	937302687	937309240.6	0.76211536
74	937302687	11011111011110000110101001111	937302687	937302687	0.762008797
75	937302687	11011111011110000110101001111	937040543	937276472.6	0.762008797
76	937302687	11011111011110000110101001111	937302687	937302687	0.762008797
77	937319071	11011111011110010110101001111	937302687	937304325.4	0.762035437
78	937319071	11011111011110010110101001111	937302686	937304331.7	0.762035437
79	937302751	11011111011110000110101011111	937302686	937302693.3	0.762008901
80	937302751	11011111011110000110101011111	937302687	937302699.8	0.762008901
81	937302751	11011111011110000110101011111	937302687	937302699.8	0.762008901
82	939399839	11011111111110000110101001111	937302687	937512415	0.765422499
83	937302751	11011111011110000110101011111	903748255	933947263	0.762008901
84	937302751	11011111011110000110101011111	903748255	933947263	0.762008901
85	937302751	11011111011110000110101011111	937302687	937302706.2	0.762008901
86	937302751	11011111011110000110101011111	928914143	936463845.4	0.762008901

87	937335519	11011111011110100110101 1011111	928914079	936467128.6	0.762062181
88	937335519	11011111011110100110101 1011111	928914079	936470411.4	0.762062181
89	937302751	11011111011110000110101 1011111	928914079	934786111	0.762008901
90	937302751	11011111011110000110101 1011111	928914079	934786111	0.762008901
91	937302751	11011111011110000110101 1011111	928914079	936463845.4	0.762008901
92	937302751	11011111011110000110101 1011111	928914079	936358987.8	0.762008901
93	937302687	11011111011110000110101 0011111	928914079	935520114.2	0.762008797
94	937302687	11011111011110000110101 0011111	928914079	935624965.4	0.762008797
95	937302687	11011111011110000110101 0011111	928914079	935624965.4	0.762008797
96	937302687	11011111011110000110101 0011111	928914079	936463826.2	0.762008797
97	937302687	11011111011110000110101 0011111	928914079	935624965.4	0.762008797
98	937302687	11011111011110000110101 0011111	920525471	934786104.6	0.762008797
99	937368223	11011111011111000110101 0011111	920525471	933953797.4	0.76211536
100	937368223	11011111011111000110101 0011111	920525471	933973458.2	0.76211536

El mejor valor de x entre todas las generaciones fue 1071520415, y su equivalente en binario es 11111111011110000110101 0011111.

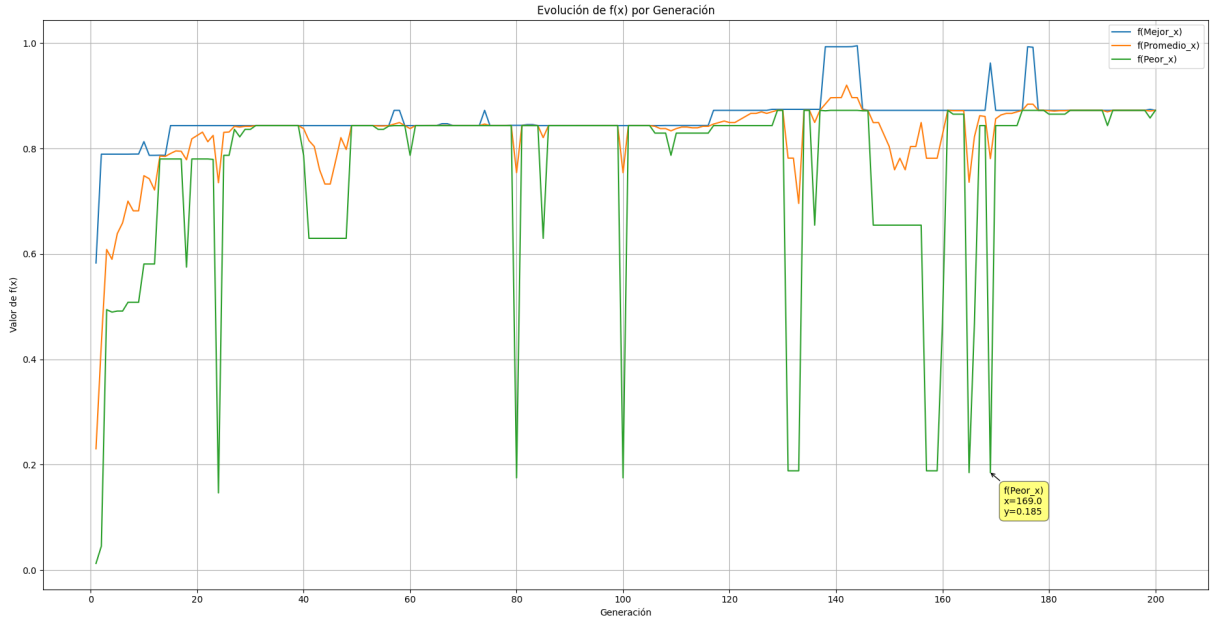

Figura 3: 200 generaciones - Método de Ruleta

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	819814888	11000011011101011 0000111101000	121337975	515346213.7	0.582950747
2	954032405	11100011011101011 0000100010101	228634153	703518165.2	0.789453426
3	954032257	11100011011101011 0000010000001	754924161	837637321.3	0.789453181
4	954032260	11100011011101011 0000010000100	751380139	824697828.9	0.789453186
5	954032257	11100011011101011 0000010000001	752920597	858120826.3	0.789453181

6	954032257	11100011011101011000001000001	752920597	871829221.2	0.789453181
7	954032661	11100011011101011000100010101	765504021	898534179.6	0.789453849
8	954143873	11100011011111000101001000001	765540993	886671201.2	0.789637914
9	954163733	11100011011111011000100010101	765536789	886671241.3	0.789670786
10	968309377	11100110110111001110101000001	818461313	928979318.8	0.813258359
11	952711812	11100011001001001110101000100	818461313	925437149.2	0.78726938
12	952712849	11100011001001001111101001001	818461313	912014443.3	0.787271094
13	952842881	11100011001011001110101000001	948508289	951883382.1	0.787486012
14	952712849	11100011001001001111101001001	948508289	951451778.9	0.787271094
15	986266241	11101011001001001110101000001	948517505	954808144.3	0.843701064
16	986266244	11101011001001001110101000100	948517505	957744974.7	0.843701069
17	986266241	11101011001001001110101000001	948517505	957325492.8	0.843701064
18	986266244	11101011001001001110101000100	814299780	947626166.4	0.843701069
19	986266244	11101011001001001110101000100	948517505	971481269.1	0.843701069
20	986266244	11101011001001001110101000100	948517505	975256143	0.843701069
21	986266244	11101011001001001110101000100	948517505	978978587.8	0.843701069
22	986266244	11101011001001001110101000100	948516993	968178203.2	0.843701069
23	986282113	11101011001001011110001000001	947993217	975310209.9	0.843728219
24	986282113	11101011001001011110001000001	411122305	920733467.9	0.843728219
25	986266245	11101011001001001110101000101	952728196	978719772.8	0.843701071
26	986266244	11101011001001001110101000100	952728193	979139203.5	0.843701069
27	986266245	11101011001001001110101000101	982071940	985427383.1	0.843701071
28	986266245	11101011001001001110101000101	973683329	984588522.3	0.843701071
29	986266247	11101011001001001110101000111	982071940	985427383.5	0.843701074
30	986266247	11101011001001001110101000111	982071940	985427384.1	0.843701074
31	986266247	11101011001001001110101000111	986266244	986266244.9	0.843701074
32	986266247	11101011001001001110101000111	986266244	986266244.9	0.843701074
33	986266247	11101011001001001110101000111	986262148	986265834.7	0.843701074
34	986266247	11101011001001001110101000111	986262148	986265835	0.843701074
35	986266311	11101011001001001110101100111	986266244	986266250.7	0.843701184
36	986282628	11101011001001011110101000100	986266244	986267882.4	0.843729101
37	986282628	11101011001001011110101000100	986266244	986267882.4	0.843729101
38	986266244	11101011001001001110101000100	986266244	986266244	0.843701069
39	986266244	11101011001001001110101000100	986266244	986266244	0.843701069
40	986266244	11101011001001001110101000100	952711812	982910800.8	0.843701069
41	986266244	11101011001001001110101000100	852048516	969489028	0.843701069
42	986266244	11101011001001001110101000100	852048516	962778141.6	0.843701069
43	986266244	11101011001001001110101000100	852048516	935934596	0.843701069
44	986266244	11101011001001001110101000100	851982980	919150826.4	0.843701069
45	986266244	11101011001001001110101000100	851982980	919137719.2	0.843701069
46	986266244	11101011001001001110101000100	852048516	945987818.4	0.843701069
47	986266244	11101011001001001110101000100	852048516	972831364	0.843701069
48	986266244	11101011001001001110101000100	852048516	959409591.2	0.843701069

49	986266244	11101011001001001111010100000100	986200708	986253136.8	0.843701069
50	986266244	11101011001001001111010100000100	986200708	986253136.8	0.843701069
51	986266244	11101011001001001111010100000100	986200708	986246583.2	0.843701069
52	986266244	11101011001001001111010100000100	986200708	986259690.4	0.843701069
53	986266244	11101011001001001111010100000100	986200708	986259690.4	0.843701069
54	986266260	11101011001001001111010100010100	982071940	985841900	0.843701096
55	986282628	11101011001001011111010100000100	982071940	985838623.2	0.843729101
56	986282628	11101011001001011111010100000100	985741956	986205624	0.843729101
57	1003043460	11101111001001001111010100000100	986217092	987939050.4	0.872649335
58	1003043460	11101111001001001111010100000100	986217092	989611856.8	0.872649335
59	986266244	11101011001001001111010100000100	986217092	986251498.4	0.843701069
60	986266244	11101011001001001111010100000100	952711812	982891143.2	0.843701069
61	986266276	11101011001001001111010101000100	986217092	986256416.8	0.843701124
62	986282660	11101011001001011111010101000100	986200964	986253165.6	0.843729155
63	986282660	11101011001001011111010101000100	986266244	986271165.2	0.843729155
64	986282660	11101011001001011111010101000100	986266240	986272799.6	0.843729155
65	986282628	11101011001001011111010100000100	986266240	986276079.2	0.843729101
66	988363392	11101011101001001111010100000000	986266240	986485794	0.847292893
67	988363392	11101011101001001111010100000000	986266244	986487439.6	0.847292893
68	986282660	11101011001001011111010101000100	986266244	986279366	0.843729155
69	986282660	11101011001001011111010101000100	986266244	986279369.6	0.843729155
70	986282660	11101011001001011111010101000100	986266276	986277734.4	0.843729155
71	986282660	11101011001001011111010101000100	986266276	986276195.2	0.843729155
72	986282660	11101011001001011111010101000100	986266276	986277933.2	0.843729155
73	986282660	11101011001001011111010101000100	986267268	986278028.4	0.843729155
74	1003059840	11101111001001011111010100000000	986267264	987955745.6	0.872677837
75	986282688	11101011001001011111010110000000	986267268	986277218.8	0.843729203
76	986282688	11101011001001011111010110000000	986267300	986276402	0.843729203
77	986282624	11101011001001011111010100000000	986267300	986274856	0.843729094
78	986300068	11101011001001101111111010100100	986267296	986276600.4	0.843758939
79	986562212	11101011001101101111111010100100	986267296	986308655.2	0.844207516
80	986562212	11101011001101101111111010100100	449396384	932613688	0.844207516
81	986562212	11101011001101101111111010100100	986265248	986335906	0.844207516
82	987348644	11101011011001101111111010100100	986265252	986388334.8	0.845553962
83	987348644	11101011011001101111111010100100	986201760	986375432	0.845553962
84	986300068	11101011001001101111111010100100	986201760	986254190.8	0.843758939
85	986300068	11101011001001101111111010100100	852082336	972825870	0.843758939
86	986300068	11101011001001101111111010100100	986201764	986260751.2	0.843758939
87	986300068	11101011001001101111111010100100	986201760	986270576.8	0.843758939
88	986300068	11101011001001101111111010100100	986201760	986267299.2	0.843758939
89	986300068	11101011001001101111111010100100	986201760	986267299.2	0.843758939
90	986300068	11101011001001101111111010100100	986201760	986260745.2	0.843758939
91	986300068	11101011001001101111111010100100	986201760	986277129.6	0.843758939

92	986300068	11101011001001101111010100100	986267296	986283683.2	0.843758939
93	986300068	11101011001001101111010100100	986267296	986273852.4	0.843758939
94	986300068	11101011001001101111010100100	986267296	986273852	0.843758939
95	986267300	11101011001001001111010100100	986267296	986267297.6	0.843702876
96	986267300	11101011001001001111010100100	986267296	986267297.2	0.843702876
97	986267302	11101011001001001111010100110	986265252	986267092.2	0.843702879
98	986283680	11101011001001011111010100000	986265252	986268730.6	0.8437309
99	986283680	11101011001001011111010100000	986267296	986270572.8	0.8437309
100	986267300	11101011001001001111010100100	449396384	932580205.2	0.843702876
101	986267297	11101011001001001111010100001	986267296	986267296.1	0.843702871
102	986267297	11101011001001001111010100001	986267296	986267296.2	0.843702871
103	986267297	11101011001001001111010100001	986267296	986267296.2	0.843702871
104	986267297	11101011001001001111010100001	986267296	986267296.3	0.843702871
105	986267360	11101011001001001111011100000	986267296	986267302.5	0.843702978
106	986267360	11101011001001001111011100000	977878688	985428441.8	0.843702978
107	986267361	11101011001001001111011100001	977878688	982911859.5	0.84370298
108	986398368	11101011001011001111010100000	977878688	982924974.6	0.843927135
109	986398368	11101011001011001111010100000	952712945	980408406.8	0.843927135
110	986398368	11101011001011001111010100000	977878688	982926606.5	0.843927135
111	986398369	11101011001011001111010100001	977878688	984604320.1	0.843927137
112	986398369	11101011001011001111010100001	977878688	984617427.4	0.843927137
113	986398369	11101011001011001111010100001	977878688	983778566.6	0.843927137
114	986398369	11101011001011001111010100001	977878689	983791469.1	0.843927137
115	986398368	11101011001011001111010100000	977878689	985444615.3	0.843927135
116	986398368	11101011001011001111010100000	977878688	985444820.9	0.843927135
117	1003044512	11101111001001001111010100000	986267296	987946659.2	0.872651166
118	1003044512	11101111001001001111010100000	986267296	989624376.8	0.872651166
119	1003044512	11101111001001001111010100000	986267296	991302094.4	0.872651166
120	1003044512	11101111001001001111010100000	986267296	989627656.8	0.872651166
121	1003044520	11101111001001001111010101000	986267296	989626020	0.87265118
122	1003044520	11101111001001001111010101000	986267296	992984740	0.87265118
123	1003060904	11101111001001011111010101000	986267304	996340185.6	0.872679688
124	1003060904	11101111001001011111010101000	986267304	999700542.4	0.872679688
125	1003060904	11101111001001011111010101000	986283680	999712011.2	0.872679688
126	1003060904	11101111001001011111010101000	986283680	1001378263	0.872679688
127	1003060904	11101111001001011111010101000	986283680	999698903.2	0.872679688
128	1004093096	11101111011001001111010101000	986267304	1001478207	0.874476661
129	1004093096	11101111011001001111010101000	1003044512	1003155928	0.874476661
130	1004093096	11101111011001001111010101000	1003044512	1003157563	0.874476661
131	1004093096	11101111011001001111010101000	466173600	949481940.8	0.874476661
132	1004093088	11101111011001001111010100000	466173608	949493409	0.874476647
133	1004093088	11101111011001001111010100000	466173600	895793211.4	0.874476647
134	1004093088	11101111011001001111010100000	1003044520	1003182965	0.874476647

135	1004093088	11101111011001001111010100000	1003044520	1003167402	0.874476647
136	1004093096	11101111011001001111010101000	868826792	989743992	0.874476661
137	1004093096	11101111011001001111010101000	1003044512	1003268979	0.874476661
138	1070153376	11111111001001001111010100000	1002536616	1009915967	0.993327166
139	1070153384	11111111001001001111010101000	1003044512	1016690751	0.99332718
140	1070153384	11111111001001001111010101000	1003044520	1016807080	0.99332718
141	1070153384	11111111001001001111010101000	1003044520	1016833294	0.99332718
142	1070153400	11111111001001001111010111000	1003044520	1030123997	0.99332721
143	1070284456	11111111001011001111010101000	1003044520	1016820189	0.99357052
144	1071201960	11111111011001001111010101000	1003044520	1016807083	0.995274732
145	1004224168	11101111011011001111010101000	1002520232	1003110061	0.87470498
146	1003044536	11101111001001001111010111000	1002520224	1002939665	0.872651207
147	1003044536	11101111001001001111010111000	868826792	989517889.6	0.872651207
148	1003044536	11101111001001001111010111000	868826792	989622750.4	0.872651207
149	1003044536	11101111001001001111010111000	868826792	976200976	0.872651207
150	1003044536	11101111001001001111010111000	868826792	962779206.4	0.872651207
151	1003044584	11101111001001001111011101000	868826792	935935668.8	0.872651291
152	1003044536	11101111001001001111010111000	868826792	949357434.4	0.872651207
153	1003044536	11101111001001001111010111000	868826792	935935664	0.872651207
154	1003044536	11101111001001001111010111000	868826792	962779212.8	0.872651207
155	1003044536	11101111001001001111010111000	868826792	962779212.8	0.872651207
156	1003044536	11101111001001001111010111000	868826792	989622756.8	0.872651207
157	1003044536	11101111001001001111010111000	466173624	949357440	0.872651207
158	1003044536	11101111001001001111010111000	466173624	949357336	0.872651207
159	1003044536	11101111001001001111010111000	466173624	949357436.8	0.872651207
160	1003044536	11101111001001001111010111000	734609064	976200977.6	0.872651207
161	1003044536	11101111001001001111010111000	1003044520	1003044525	0.872651207
162	1003044536	11101111001001001111010111000	998850216	1002625093	0.872651207
163	1003044520	11101111001001001111010101000	998850232	1002625091	0.87265118
164	1003044520	11101111001001001111010101000	998850232	1002625090	0.87265118
165	1003044520	11101111001001001111010101000	461979320	921255622.4	0.87265118
166	1003044520	11101111001001001111010101000	734609080	973684420.8	0.87265118
167	1003044520	11101111001001001111010101000	986267304	997172545.6	0.87265118
168	1003044520	11101111001001001111010101000	986267304	996333710.4	0.87265118
169	1053376168	11111011001001001111010101000	461979304	948938075.2	0.962425758
170	1003044520	11101111001001001111010101000	986267304	993817102.4	0.87265118
171	1003044520	11101111001001001111010101000	986267304	998011355.2	0.87265118
172	1003044520	11101111001001001111010101000	986267304	999689076.8	0.87265118
173	1003044520	11101111001001001111010101000	986267304	999689076.8	0.87265118
174	1003044520	11101111001001001111010101000	986267304	1001366798	0.87265118
175	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
176	1070153384	11111111001001001111010101000	1003044520	1009755406	0.99332718
177	1069629096	11111111000001001111010101000	1003044520	1009702978	0.99235412

178	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
179	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
180	1003044776	11101111001001001111110101000	998850216	1002624296	0.872651625
181	1003044520	11101111001001001111010101000	998850216	1002205659	0.87265118
182	1003044520	11101111001001001111010101000	998850216	1002625090	0.87265118
183	1003077288	11101111001001101111010101000	998850216	1002628366	0.872708197
184	1003077288	11101111001001101111010101000	1003044520	1003051074	0.872708197
185	1003077288	11101111001001101111010101000	1003044520	1003051074	0.872708197
186	1003077288	11101111001001101111010101000	1003044520	1003051074	0.872708197
187	1003077288	11101111001001101111010101000	1003044488	1003047794	0.872708197
188	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
189	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
190	1003044520	11101111001001001111010101000	1003044520	1003044520	0.87265118
191	1003044520	11101111001001001111010101000	986267304	1001366786	0.87265118
192	1003044520	11101111001001001111010101000	1003044392	1003044494	0.87265118
193	1003044520	11101111001001001111010101000	1003044392	1003044456	0.87265118
194	1003044776	11101111001001001111110101000	1003044392	1003044494	0.872651625
195	1003044776	11101111001001001111110101000	1003044392	1003044482	0.872651625
196	1003044520	11101111001001001111010101000	1003044392	1003044456	0.87265118
197	1003044520	11101111001001001111010101000	1003044392	1003044482	0.87265118
198	1003044520	11101111001001001111010101000	1003044392	1003044469	0.87265118
199	1004092968	11101111011001001111000101000	994655912	1002310472	0.874476438
200	1003044536	11101111001001001111010111000	1003044392	1003044458	0.872651207

El mejor valor de x entre todas las generaciones fue 1071201960, y su equivalente en binario es 11111111011001001111010101000.

6.2. Método de Torneo

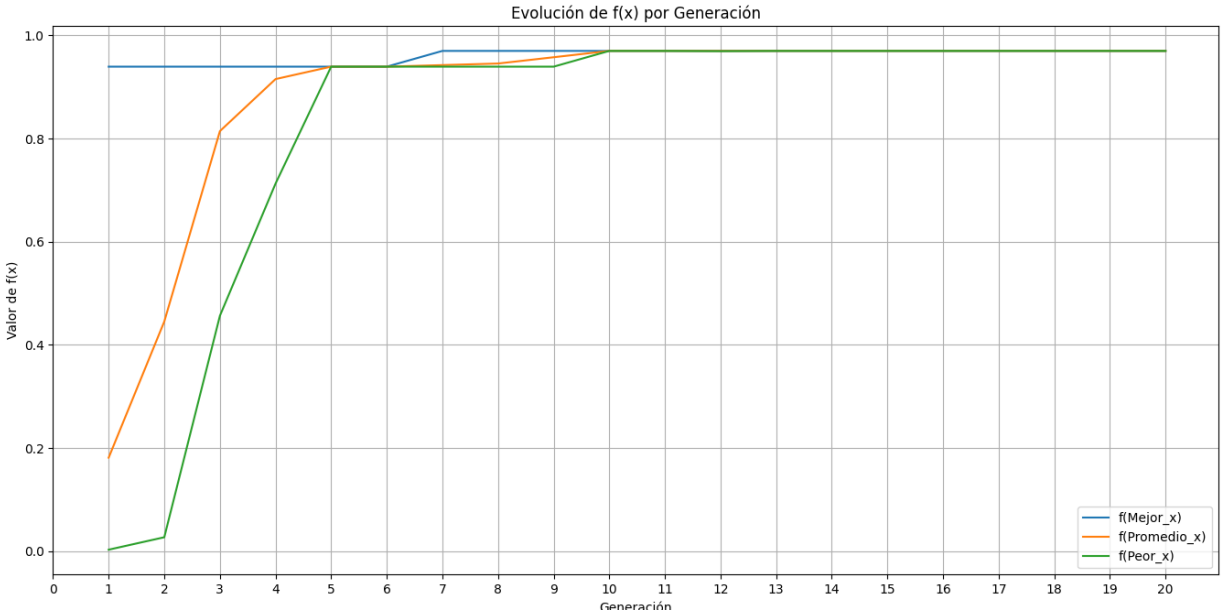

Figura 4: 20 generaciones - Método de Torneo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	1040694419	11111000000111101111 00010010011	60319524	457404601.8	0.939391686
2	1040694429	11111000000111101111 00010011101	177293823	716591575.6	0.939391704
3	1040694504	11111000000111101111 00011101000	725510685	969008540.2	0.939391839
4	1040694504	11111000000111101111 00011101000	906476701	1027272663	0.939391839
5	1040694504	11111000000111101111 00011101000	1040694429	1040694459	0.939391839
6	1040694504	11111000000111101111 00011101000	1040694504	1040694504	0.939391839
7	1057471720	11111100000111101111 00011101000	1040694504	1042372226	0.969924176
8	1057471720	11111100000111101111 00011101000	1040694504	1044049947	0.969924176
9	1057471720	11111100000111101111 00011101000	1040694504	1050760834	0.969924176
10	1057471722	11111100000111101111 00011101010	1057467624	1057471311	0.96992418
11	1057471722	11111100000111101111 00011101010	1057471720	1057471721	0.96992418
12	1057471722	11111100000111101111 00011101010	1057209578	1057445507	0.96992418
13	1057471722	11111100000111101111 00011101010	1057471720	1057471722	0.96992418
14	1057471722	11111100000111101111 00011101010	1057471722	1057471722	0.96992418
15	1057471978	11111100000111101111 01111101010	1057471722	1057471748	0.96992465
16	1057471978	11111100000111101111 01111101010	1057471722	1057471824	0.96992465
17	1057471978	11111100000111101111 01111101010	1057471722	1057471927	0.96992465
18	1057471978	11111100000111101111 01111101010	1057471978	1057471978	0.96992465
19	1057471978	11111100000111101111 01111101010	1057471978	1057471978	0.96992465
20	1057471978	11111100000111101111 01111101010	1057471978	1057471978	0.96992465

El mejor valor de x entre todas las generaciones fue 1057471978, y su equivalente en binario es 11111100000111101111 01111101010.

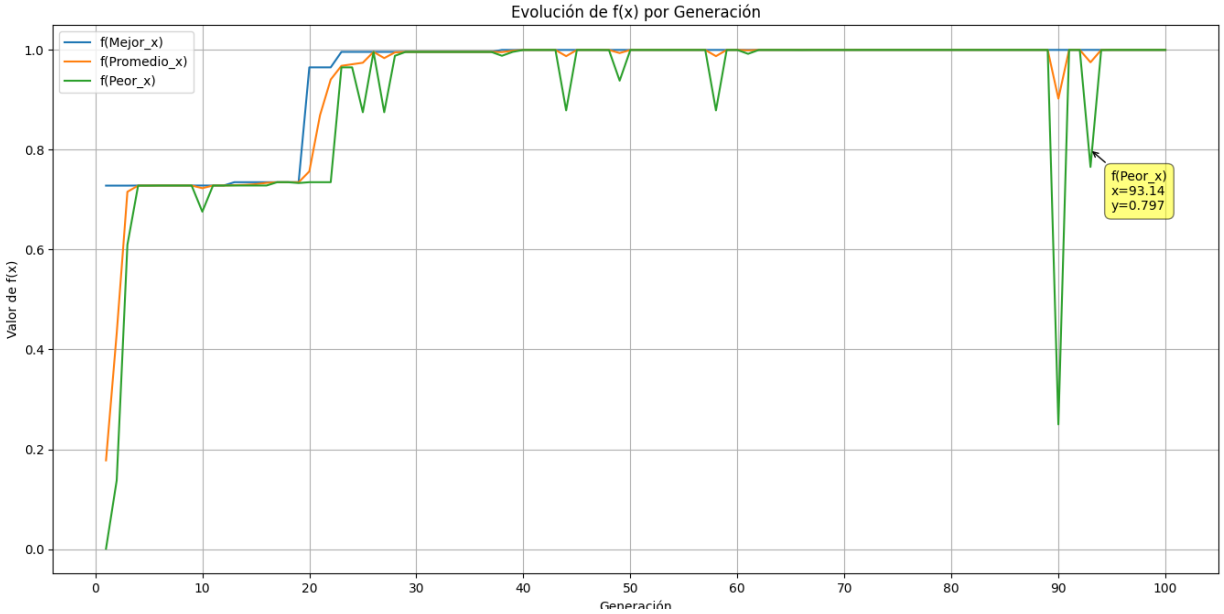

Figura 5: 100 generaciones - Método de Torneo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	916019590	11011010011001010111001110000110	33265439	452725010.4	0.727796201
2	916019590	11011010011001010111001110000110	398344653	705866973.7	0.727796201
3	916019590	11011010011001010111001110000110	838195590	908235763.5	0.727796201
4	916150662	11011010011011010111001110000110	916017542	916032492.4	0.728004494
5	916150662	11011010011011010111001110000110	916019590	916098233.2	0.728004494
6	916150678	11011010011011010111001110010110	916150662	916150663.6	0.72800452
7	916150678	11011010011011010111001110010110	916150662	916150671.6	0.72800452
8	916150678	11011010011011010111001110010110	916150678	916150678	0.72800452
9	916150678	11011010011011010111001110010110	916150678	916150678	0.72800452
10	916150678	11011010011011010111001110010110	882596246	912795234.8	0.72800452
11	916150678	11011010011011010111001110010110	916150678	916150678	0.72800452
12	916150678	11011010011011010111001110010110	916150678	916150678	0.72800452
13	920344982	11011011011011010111001110010110	916150678	916570108.4	0.734685652
14	920344982	11011011011011010111001110010110	916150678	916989538.8	0.734685652
15	920344982	11011011011011010111001110010110	916150678	917828399.6	0.734685652
16	920344982	11011011011011010111001110010110	916150678	919506121.2	0.734685652
17	920344982	11011011011011010111001110010110	920344982	920344982	0.734685652
18	920344982	11011011011011010111001110010110	920344982	920344982	0.734685652
19	920344982	11011011011011010111001110010110	919296406	920240124.4	0.734685652
20	1054562710	11111011011011010111001110010110	920344982	933766754.8	0.964595167
21	1054562710	11111011011011010111001110010110	920344982	1000875619	0.964595167
22	1054562710	11111011011011010111001110010110	920344982	1041140937	0.964595167
23	1071339926	11111111011011010111001110010110	1054562710	1056240432	0.995531122
24	1071339926	11111111011011010111001110010110	1054562710	1057918153	0.995531122

25	1071339926	11111111011011010110011001 0110	1004231062	1059595875	0.995531122
26	1071339926	11111111011011010110011001 0110	1071337878	1071339708	0.995531122
27	1071339926	11111111011011010110011001 0110	1004231062	1064629040	0.995531122
28	1071339926	11111111011011010110011001 0110	1067145622	1070815638	0.995531122
29	1071339926	11111111011011010110011001 0110	1071339922	1071339926	0.995531122
30	1071339926	11111111011011010110011001 0110	1071335830	1071339516	0.995531122
31	1071339926	11111111011011010110011001 0110	1071339926	1071339926	0.995531122
32	1071339926	11111111011011010110011001 0110	1071339926	1071339926	0.995531122
33	1071339958	11111111011011010110011011 0110	1071339926	1071339929	0.995531182
34	1071339958	11111111011011010110011011 0110	1071339926	1071339936	0.995531182
35	1071339958	11111111011011010110011011 0110	1071339926	1071339946	0.995531182
36	1071348150	11111111011011011110011011 0110	1071339942	1071340776	0.995546406
37	1071348150	11111111011011011110011011 0110	1071339958	1071340777	0.995546406
38	1073445302	11111111111011011110011011 0110	1067145654	1071131881	0.999447763
39	1073445302	11111111111011011110011011 0110	1071339958	1072603161	0.999447763
40	1073445302	11111111111011011110011011 0110	1073445270	1073445299	0.999447763
41	1073445302	11111111111011011110011011 0110	1073443254	1073445097	0.999447763
42	1073445302	11111111111011011110011011 0110	1073445302	1073445302	0.999447763
43	1073445302	11111111111011011110011011 0110	1073445302	1073445302	0.999447763
44	1073445302	11111111111011011110011011 0110	1006336438	1066734416	0.999447763
45	1073445302	11111111111011011110011011 0110	1073445302	1073445302	0.999447763
46	1073446326	11111111111011011111011011 0110	1073445302	1073445404	0.99944967
47	1073446326	11111111111011011111011011 0110	1073445302	1073445507	0.99944967
48	1073446326	11111111111011011111011011 0110	1073314230	1073432809	0.99944967
49	1073446326	11111111111011011111011011 0110	1039891894	1070090883	0.99944967
50	1073446326	11111111111011011111011011 0110	1073446326	1073446326	0.99944967
51	1073446326	11111111111011011111011011 0110	1073446326	1073446326	0.99944967
52	1073446326	11111111111011011111011011 0110	1073446326	1073446326	0.99944967
53	1073446326	11111111111011011111011011 0110	1073429942	1073444688	0.99944967
54	1073446326	11111111111011011111011011 0110	1073446326	1073446326	0.99944967
55	1073446326	11111111111011011111011011 0110	1073445302	1073446224	0.99944967
56	1073446326	11111111111011011111011011 0110	1073446326	1073446326	0.99944967
57	1073446334	11111111111011011111011011 1110	1073446326	1073446327	0.999449685
58	1073446334	11111111111011011111011011 1110	1006337462	1066734623	0.999449685
59	1073446334	11111111111011011111011011 1110	1073429950	1073444693	0.999449685
60	1073446334	11111111111011011111011011 1110	1073446318	1073446332	0.999449685
61	1073446334	11111111111011011111011011 1110	1069252030	1073026904	0.999449685
62	1073446334	11111111111011011111011011 1110	1073445310	1073446232	0.999449685
63	1073446334	11111111111011011111011011 1110	1073446318	1073446332	0.999449685
64	1073446334	11111111111011011111011011 1110	1073444286	1073446129	0.999449685
65	1073446334	11111111111011011111011011 1110	1073444286	1073446129	0.999449685
66	1073446334	11111111111011011111011011 1110	1073380798	1073439780	0.999449685
67	1073446334	11111111111011011111011011 1110	1073446334	1073446334	0.999449685

68	1073446334	11111111111011011110110111110	1073446334	1073446334	0.999449685
69	1073479102	11111111111011111110110111110	1073446334	1073449611	0.999510704
70	1073479102	11111111111011111110110111110	1073446334	1073465995	0.999510704
71	1073479102	11111111111011111110110111110	1073446302	1073472545	0.999510704
72	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
73	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
74	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
75	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
76	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
77	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
78	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
79	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
80	1073479102	11111111111011111110110111110	1073478846	1073479076	0.999510704
81	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
82	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
83	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
84	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
85	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
86	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
87	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
88	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
89	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
90	1073479102	11111111111011111110110111110	536608190	1019792011	0.999510704
91	1073479102	11111111111011111110110111110	1073479102	1073479102	0.999510704
92	1073479102	11111111111011111110110111110	1073446334	1073475825	0.999510704
93	1073479102	11111111111011111110110111110	939261374	1060057326	0.999510704
94	1073479102	11111111111011111110110111110	1073348030	1073465993	0.999510704
95	1073479614	11111111111011111111110111110	1073479102	1073479153	0.999511657
96	1073479614	11111111111011111111110111110	1073479102	1073479307	0.999511657
97	1073479614	11111111111011111111110111110	1073479102	1073479512	0.999511657
98	1073479614	11111111111011111111110111110	1073479614	1073479614	0.999511657
99	1073479614	11111111111011111111110111110	1073479614	1073479614	0.999511657
100	1073479614	11111111111011111111110111110	1073478590	1073479512	0.999511657

El mejor valor de x entre todas las generaciones fue 1073479614, y su equivalente en binario es 11111111111011111111110111110.

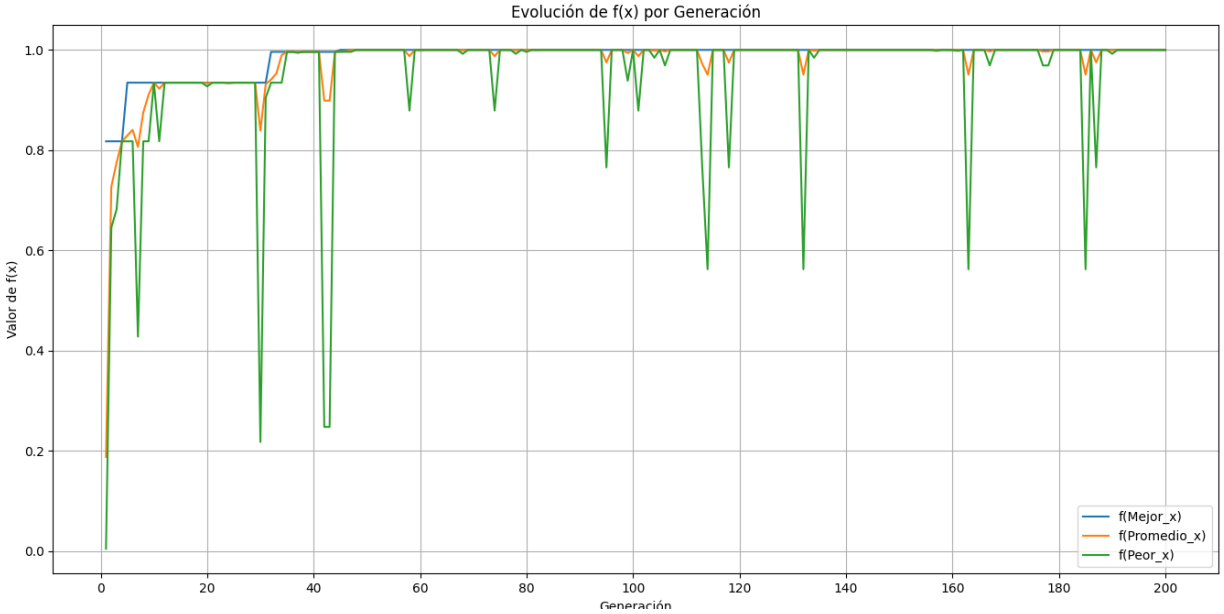

Figura 6: 200 generaciones - Método de Torneo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	970905563	11100111011110110101111011011	77004403	465870346.1	0.817625146
2	970967321	11100111011111110010010011001	862771163	915707553	0.817729166
3	970971099	11100111011111110101111011011	886946859	945731280.6	0.817735529
4	970971099	11100111011111110101111011011	970905563	970934676.4	0.817735529
5	1038079963	11110111011111110101111011011	970971099	977681985.4	0.934677693
6	1038079963	11110111011111110101111011011	970971099	984392871.8	0.934677693
7	1038079963	11110111011111110101111011011	702535643	964260212.6	0.934677693
8	1038079963	11110111011111110101111011011	970971099	1004525531	0.934677693
9	1038079963	11110111011111110101111011011	970971099	1024658139	0.934677693
10	1038079963	11110111011111110101111011011	1038079963	1038079963	0.934677693
11	1038079963	11110111011111110101111011011	970971099	1031342862	0.934677693
12	1038079963	11110111011111110101111011011	1037948891	1038066856	0.934677693
13	1038079963	11110111011111110101111011011	1038079963	1038079963	0.934677693
14	1038079963	11110111011111110101111011011	1038079955	1038079962	0.934677693
15	1038079963	11110111011111110101111011011	1038079707	1038079937	0.934677693
16	1038079963	11110111011111110101111011011	1038079955	1038079962	0.934677693
17	1038088155	11110111011111111101111011011	1038079955	1038080781	0.934692445
18	1038088155	11110111011111111101111011011	1038079963	1038084008	0.934692445
19	1038088155	11110111011111111101111011011	1038079963	1038087182	0.934692445
20	1038088155	11110111011111111101111011011	1033893851	1037668673	0.934692445
21	1038088155	11110111011111111101111011011	1038088155	1038088155	0.934692445
22	1038088155	11110111011111111101111011011	1038088155	1038088155	0.934692445
23	1038088155	11110111011111111101111011011	1038088155	1038088155	0.934692445
24	1038088155	11110111011111111101111011011	1037563867	1038035726	0.934692445

25	1038088155	11110111011111111101111011011	1038087643	1038088104	0.934692445
26	1038088155	11110111011111111101111011011	1038055387	1038084878	0.934692445
27	1038088159	11110111011111111101111011111	1038088155	1038088155	0.934692452
28	1038088159	11110111011111111101111011111	1038088155	1038088156	0.934692452
29	1038088159	11110111011111111101111011111	1038088155	1038088157	0.934692452
30	1038088159	11110111011111111101111011111	501217247	983562206.6	0.934692452
31	1038088159	11110111011111111101111011111	1021310943	1036410437	0.934692452
32	1071642591	11111111011111111101111011111	1038088159	1041443602	0.996093698
33	1071642591	11111111011111111101111011111	1038088159	1048154489	0.996093698
34	1071642591	11111111011111111101111011111	1038088159	1068182265	0.996093698
35	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
36	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
37	1071642591	11111111011111111101111011111	1070594015	1071537733	0.996093698
38	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
39	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
40	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
41	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
42	1071642591	11111111011111111101111011111	534771679	1017948946	0.996093698
43	1071642591	11111111011111111101111011111	534771679	1017955500	0.996093698
44	1071642591	11111111011111111101111011111	1071642591	1071642591	0.996093698
45	1073739743	11111111111111111101111011111	1071641567	1071852204	0.999996126
46	1073739743	11111111111111111101111011111	1071642591	1072271737	0.999996126
47	1073739743	11111111111111111101111011111	1071642591	1073320313	0.999996126
48	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
49	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
50	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
51	1073739743	11111111111111111101111011111	1073735647	1073739333	0.999996126
52	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
53	1073739743	11111111111111111101111011111	1073731551	1073738924	0.999996126
54	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
55	1073739743	11111111111111111101111011111	1073738719	1073739641	0.999996126
56	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
57	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
58	1073739743	11111111111111111101111011111	1006630879	1067028857	0.999996126
59	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
60	1073739743	11111111111111111101111011111	1073477599	1073713529	0.999996126
61	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
62	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
63	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
64	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
65	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
66	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126
67	1073739743	11111111111111111101111011111	1073739743	1073739743	0.999996126

68	1073739743	11111111111111111011111011111	1069545439	1073320313	0.999996126
69	1073739743	11111111111111111011111011111	1073739743	1073739743	0.999996126
70	1073739743	11111111111111111011111011111	1073739743	1073739743	0.999996126
71	1073741791	11111111111111111111111011111	1073739743	1073739948	0.99999994
72	1073741791	11111111111111111111111011111	1073739743	1073740357	0.99999994
73	1073741791	11111111111111111111111011111	1073739743	1073741381	0.99999994
74	1073741791	11111111111111111111111011111	1006632927	1067030905	0.99999994
75	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
76	1073741791	11111111111111111111111011111	1073740767	1073741689	0.99999994
77	1073741791	11111111111111111111111011111	1073610719	1073728684	0.99999994
78	1073741791	11111111111111111111111011111	1069547487	1073322361	0.99999994
79	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
80	1073741791	11111111111111111111111011111	1071644639	1073532076	0.99999994
81	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
82	1073741791	11111111111111111111111011111	1073610719	1073728684	0.99999994
83	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
84	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
85	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
86	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
87	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
88	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
89	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
90	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
91	1073741791	11111111111111111111111011111	1073737695	1073741381	0.99999994
92	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
93	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
94	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
95	1073741791	11111111111111111111111011111	939524063	1060320018	0.99999994
96	1073741791	11111111111111111111111011111	1073709023	1073738501	0.99999994
97	1073741791	11111111111111111111111011111	1073741775	1073741789	0.99999994
98	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
99	1073741791	11111111111111111111111011111	1040187359	1070386348	0.99999994
100	1073741791	11111111111111111111111011111	1073610719	1073728684	0.99999994
101	1073741791	11111111111111111111111011111	1006632927	1067030890	0.99999994
102	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
103	1073741791	11111111111111111111111011111	1073740767	1073741689	0.99999994
104	1073741791	11111111111111111111111011111	1065353183	1072902930	0.99999994
105	1073741791	11111111111111111111111011111	1073741787	1073741791	0.99999994
106	1073741791	11111111111111111111111011111	1056964575	1072064069	0.99999994
107	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
108	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
109	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
110	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994

111	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
112	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
113	1073741791	11111111111111111111111011111	939524063	1058642297	0.99999994
114	1073741791	11111111111111111111111011111	805306335	1046898245	0.99999994
115	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
116	1073741791	11111111111111111111111011111	1073676255	1073735237	0.99999994
117	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
118	1073741791	11111111111111111111111011111	939524063	1060215154	0.99999994
119	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
120	1073741791	11111111111111111111111011111	1073733599	1073740972	0.99999994
121	1073741791	11111111111111111111111011111	1073741791	1073741791	0.99999994
122	1073741823	11111111111111111111111111111	1073741791	1073741794	1
123	1073741823	11111111111111111111111111111	1073741791	1073741801	1
124	1073741823	11111111111111111111111111111	1073741791	1073741820	1
125	1073741823	11111111111111111111111111111	1073741823	1073741823	1
126	1073741823	11111111111111111111111111111	1073741823	1073741823	1
127	1073741823	11111111111111111111111111111	1073741823	1073741823	1
128	1073741823	11111111111111111111111111111	1073733631	1073741004	1
129	1073741823	11111111111111111111111111111	1073741823	1073741823	1
130	1073741823	11111111111111111111111111111	1073741823	1073741823	1
131	1073741823	11111111111111111111111111111	1073741823	1073741823	1
132	1073741823	11111111111111111111111111111	805306367	1046898277	1
133	1073741823	11111111111111111111111111111	1073741815	1073741822	1
134	1073741823	11111111111111111111111111111	1065353215	1072902962	1
135	1073741823	11111111111111111111111111111	1073741695	1073741810	1
136	1073741823	11111111111111111111111111111	1073741821	1073741823	1
137	1073741823	11111111111111111111111111111	1073741823	1073741823	1
138	1073741823	11111111111111111111111111111	1073741823	1073741823	1
139	1073741823	11111111111111111111111111111	1073741823	1073741823	1
140	1073741823	11111111111111111111111111111	1073741823	1073741823	1
141	1073741823	11111111111111111111111111111	1073741823	1073741823	1
142	1073741823	11111111111111111111111111111	1073741823	1073741823	1
143	1073741823	11111111111111111111111111111	1073741823	1073741823	1
144	1073741823	11111111111111111111111111111	1073741823	1073741823	1
145	1073741823	11111111111111111111111111111	1073741823	1073741823	1
146	1073741823	11111111111111111111111111111	1073741791	1073741820	1
147	1073741823	11111111111111111111111111111	1073741823	1073741823	1
148	1073741823	11111111111111111111111111111	1073741823	1073741823	1
149	1073741823	11111111111111111111111111111	1073741823	1073741823	1
150	1073741823	11111111111111111111111111111	1073741823	1073741823	1
151	1073741823	11111111111111111111111111111	1073741823	1073741823	1
152	1073741823	11111111111111111111111111111	1073739775	1073741618	1
153	1073741823	11111111111111111111111111111	1073741823	1073741823	1

154	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
155	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
156	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
157	1073741823	11111111111111111111111111111111	1072693247	1073636965	1
158	1073741823	11111111111111111111111111111111	1073741759	1073741816	1
159	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
160	1073741823	11111111111111111111111111111111	1073479679	1073715557	1
161	1073741823	11111111111111111111111111111111	1072693247	1073636965	1
162	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
163	1073741823	11111111111111111111111111111111	805306367	1046898277	1
164	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
165	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
166	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
167	1073741823	11111111111111111111111111111111	1056964607	1072064101	1
168	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
169	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
170	1073741823	11111111111111111111111111111111	1073610751	1073722162	1
171	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
172	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
173	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
174	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
175	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
176	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
177	1073741823	11111111111111111111111111111111	1056964607	1072064095	1
178	1073741823	11111111111111111111111111111111	1056964607	1072064101	1
179	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
180	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
181	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
182	1073741823	11111111111111111111111111111111	1073741821	1073741823	1
183	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
184	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
185	1073741823	11111111111111111111111111111111	805306367	1046898277	1
186	1073741823	11111111111111111111111111111111	1073741815	1073741822	1
187	1073741823	11111111111111111111111111111111	939524095	1060320049	1
188	1073741823	11111111111111111111111111111111	1073741759	1073741817	1
189	1073741823	11111111111111111111111111111111	1073741807	1073741821	1
190	1073741823	11111111111111111111111111111111	1069547519	1073322393	1
191	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
192	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
193	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
194	1073741823	11111111111111111111111111111111	1073741791	1073741820	1
195	1073741823	11111111111111111111111111111111	1073741823	1073741823	1
196	1073741823	11111111111111111111111111111111	1073741823	1073741823	1

197	1073741823	11111111111111111111111111111	1073739775	1073741618	1
198	1073741823	11111111111111111111111111111	1073741823	1073741823	1
199	1073741823	11111111111111111111111111111	1073741823	1073741823	1
200	1073741823	11111111111111111111111111111	1073741823	1073741823	1

El mejor valor de x entre todas las generaciones fue 1073741823, y su equivalente en binario es 11111111111111111111111111111.

6.3. Método de Elitismo (aplicado a ruleta)

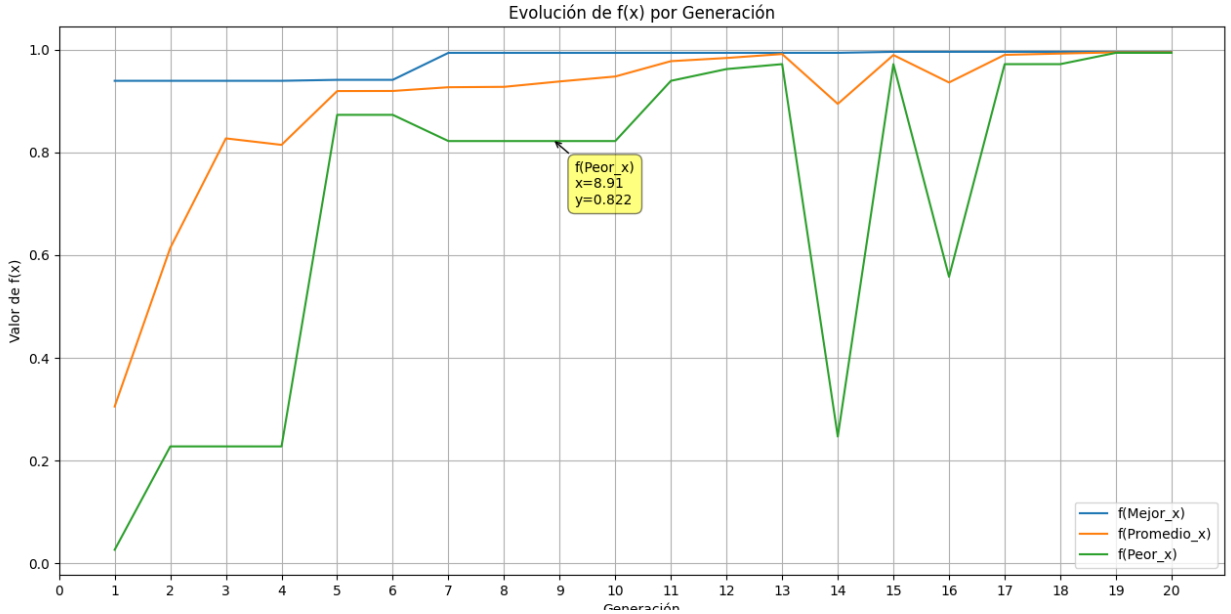

Figura 7: 20 generaciones - Método de Elitismo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	1040606582	11111000000110011001010111010110	174613749	593297339.5	0.939233119
2	1040606582	11111000000110011001010111010110	512319018	841462062.2	0.939233119
3	1040606582	11111000000110011001010111010110	512319045	976566227.3	0.939233119
4	1040606582	11111000000110011001010111010110	512304327	969091783	0.939233119
5	1041655158	11111000010110011001010111010110	1003234678	1029499851	0.941126925
6	1041655158	11111000010110011001010111010110	1003234678	1029604691	0.941126925
7	1070343542	11111111001100001001010111010110	973497543	1033656437	0.993680225
8	1070343542	11111111001100001001010111010110	973497718	1034075867	0.993680225
9	1070343542	11111111001100001001010111010110	973497671	1039918384	0.993680225
10	1070343542	11111111001100001001010111010110	973497671	1045236674	0.993680225
11	1070359926	11111111001100011001010111010110	1040590198	1061594451	0.993710646
12	1070359926	11111111001100011001010111010110	1053189495	1064943306	0.993710646
13	1070359926	11111111001100011001010111010110	1058432374	1069080301	0.993710646

14	1070359926	11111111001100011001010101110110	533472630	1015435825	0.993710646
15	1071408502	11111111011100011001010101110110	1058432374	1068038296	0.995658573
16	1071408502	11111111011100011001010101110110	801924470	1038809240	0.995658573
17	1071408502	11111111011100011001010101110110	1058415990	1068248028	0.995658573
18	1071408502	11111111011100011001010101110110	1058432374	1069586601	0.995658573
19	1071408502	11111111011100011001010101110110	1070359926	1070884214	0.995658573
20	1071408502	11111111011100011001010101110110	1070359926	1070884214	0.995658573

El mejor valor de x entre todas las generaciones fue 1071408502 , y su equivalente en binario es 11111111011100011001010101110110.

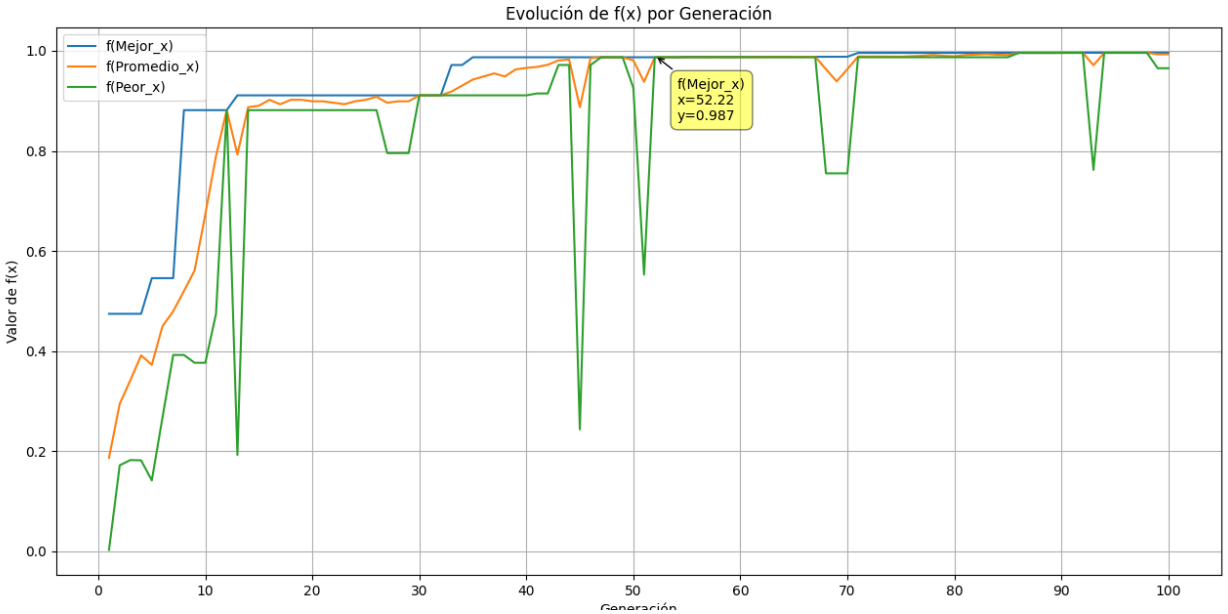

Figura 8: 100 generaciones - Método de Elitismo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	739607649	10110000010101100001000110000 1	57304084	463955072.5	0.474463763
2	739607658	10110000010101100001000110101 0	445088126	582809731.8	0.474463775
3	739617866	10110000010101101011000100101 0	458533953	627922345.5	0.474476872
4	739617866	10110000010101101011000100101 0	457672193	671838552.4	0.474476872
5	793216513	10111101000111100001100000000 1	404073473	655488777.8	0.545737446
6	793216513	10111101000111100001100000000 1	555451905	720173186.2	0.545737446
7	793216513	10111101000111100001100000000 1	672497982	743622422.9	0.545737446
8	1008053322	11110000010101101011000100101 0	672497665	774148111.6	0.88138828
9	1008053322	11110000010101101011000100101 0	658999102	804348964.3	0.88138828
10	1008053505	11110000010101101011010000000 1	658997566	881197573.8	0.8813886
11	1008053505	11110000010101101011010000000 1	739617866	954365141	0.8813886
12	1008053505	11110000010101101011010000000 1	1008043082	1008052269	0.8813886
13	1024830538	11110100010101101011000100101 0	471182410	956044025.6	0.910970634

14	1024830538	11110100010101101010110001001010	1008053322	1011408857	0.910970634
15	1024830538	11110100010101101010110001001010	1008053323	1013086597	0.910970634
16	1024830538	11110100010101101010110001001010	1008053504	1019797428	0.910970634
17	1024830721	11110100010101101010110100000001	1008053322	1014764318	0.910970959
18	1024830721	11110100010101101010110100000001	1008053505	1019797465	0.910970959
19	1024830730	11110100010101101010110100001010	1008053322	1019797503	0.910970975
20	1024830730	11110100010101101010110100001010	1008053322	1018119802	0.910970975
21	1024830730	11110100010101101010110100001010	1008053322	1018093608	0.910970975
22	1024830730	11110100010101101010110100001010	1008053322	1016442064	0.910970975
23	1024830730	11110100010101101010110100001010	1008053322	1014777469	0.910970975
24	1024830730	11110100010101101010110100001010	1008053322	1018119805	0.910970975
25	1024830730	11110100010101101010110100001010	1008053514	1019797565	0.910970975
26	1024830730	11110100010101101010110100001010	1008053514	1023153008	0.910970975
27	1024830730	11110100010101101010110100001010	957721866	1016442122	0.910970975
28	1024830730	11110100010101101010110100001010	957721866	1018119844	0.910970975
29	1024830730	11110100010101101010110100001010	957721866	1018119844	0.910970975
30	1024830730	11110100010101101010110100001010	1024830730	1024830730	0.910970975
31	1024830730	11110100010101101010110100001010	1024830474	1024830704	0.910970975
32	1024830730	11110100010101101010110100001010	1024830730	1024830730	0.910970975
33	1058385162	11111100010101101010110100001010	1024830730	1029025034	0.971600537
34	1058385162	11111100010101101010110100001010	1024830730	1035749028	0.971600537
35	1066773770	11111110010101101010110100001010	1024830730	1042446807	0.987063104
36	1066773770	11111110010101101010110100001010	1024830730	1045802250	0.987063104
37	1066773770	11111110010101101010110100001010	1024830730	1049157696	0.987063104
38	1066773770	11111110010101101010110100001010	1024830730	1045802253	0.987063104
39	1066773770	11111110010101101010110100001010	1024830730	1053561712	0.987063104
40	1066773770	11111110010101101010110100001010	1024830730	1055239434	0.987063104
41	1066773770	11111110010101101010110100001010	1026927882	1056288010	0.987063104
42	1066773770	11111110010101101010110100001010	1026927882	1058594877	0.987063104
43	1066773770	11111110010101101010110100001010	1058385162	1063418330	0.987063104
44	1066773802	11111110010101101010110100101010	1058385162	1064257191	0.987063163
45	1066773802	11111110010101101010110100101010	529902890	1011408967	0.987063163
46	1066773802	11111110010101101010110100101010	1058385162	1065934922	0.987063163
47	1066773802	11111110010101101010110100101010	1066773770	1066773799	0.987063163
48	1066773802	11111110010101101010110100101010	1066765610	1066772983	0.987063163
49	1066773802	11111110010101101010110100101010	1066741034	1066770525	0.987063163
50	1066773802	11111110010101101010110100101010	1033219370	1063418359	0.987063163
51	1066777898	11111110010101101111010100101010	798338346	1039930666	0.987070743
52	1066777898	11111110010101101111010100101010	1066773802	1066774621	0.987070743
53	1066777898	11111110010101101111010100101010	1066773802	1066775440	0.987070743
54	1066777898	11111110010101101111010100101010	1066773802	1066776260	0.987070743
55	1067302186	11111110011101101111010100101010	1066773802	1066829508	0.98804121
56	1067302186	11111110011101101111010100101010	1066773802	1066829917	0.98804121

57	1067302186	11111110011101101111010010101010	1066773802	1066881936	0.98804121
58	1067302186	11111110011101101111010010101010	1066773802	1066985975	0.98804121
59	1067302186	11111110011101101111010010101010	1066773802	1066881117	0.98804121
60	1067302186	11111110011101101111010010101010	1066773802	1066932727	0.98804121
61	1067302186	11111110011101101111010010101010	1066773802	1067038404	0.98804121
62	1067302186	11111110011101101111010010101010	1066773802	1067092061	0.98804121
63	1067302186	11111110011101101111010010101010	1066777898	1067144900	0.98804121
64	1067302186	11111110011101101111010010101010	1066777898	1067092471	0.98804121
65	1067302186	11111110011101101111010010101010	1066777898	1067197328	0.98804121
66	1067302186	11111110011101101111010010101010	1066777898	1067197328	0.98804121
67	1067302186	11111110011101101111010010101010	1066777898	1067249757	0.98804121
68	1067302186	11111110011101101111010010101010	933084458	1053827984	0.98804121
69	1067302186	11111110011101101111010010101010	933084458	1040406212	0.98804121
70	1067302186	11111110011101101111010010101010	933084458	1053775555	0.98804121
71	1071496482	11111111011101101111010010010010	1066777898	1067669187	0.995822099
72	1071496482	11111111011101101111010010010010	1066777890	1067564328	0.995822099
73	1071496482	11111111011101101111010010010010	1066777898	1067616758	0.995822099
74	1071496482	11111111011101101111010010010010	1066777898	1067511900	0.995822099
75	1071496482	11111111011101101111010010010010	1066777898	1067459523	0.995822099
76	1071496482	11111111011101101111010010010010	1066777898	1067878952	0.995822099
77	1071496490	11111111011101101111010010101010	1066777898	1068350760	0.995822114
78	1071496490	11111111011101101111010010101010	1066777890	1069294476	0.995822114
79	1071496490	11111111011101101111010010101010	1066777890	1068311437	0.995822114
80	1071496490	11111111011101101111010010101010	1066777890	1067839578	0.995822114
81	1071496490	11111111011101101111010010101010	1066777890	1069150299	0.995822114
82	1071496490	11111111011101101111010010101010	1066777898	1069621953	0.995822114
83	1071496490	11111111011101101111010010101010	1066777898	1070080707	0.995822114
84	1071496490	11111111011101101111010010101010	1066777898	1069504195	0.995822114
85	1071496490	11111111011101101111010010101010	1066777898	1069137194	0.995822114
86	1071496490	11111111011101101111010010101010	1071496490	1071496490	0.995822114
87	1071496490	11111111011101101111010010101010	1071496490	1071496490	0.995822114
88	1071496490	11111111011101101111010010101010	1071496490	1071496490	0.995822114
89	1071496490	11111111011101101111010010101010	1071463722	1071493213	0.995822114
90	1071627562	11111111011111101111010010101010	1071463722	1071506320	0.996065759
91	1071627562	11111111011111101111010010101010	1071496490	1071535812	0.996065759
92	1071627562	11111111011111101111010010101010	1071496490	1071562026	0.996065759
93	1071627562	11111111011111101111010010101010	937278762	1058153360	0.996065759
94	1071627562	11111111011111101111010010101010	1071496490	1071562026	0.996065759
95	1071627562	11111111011111101111010010101010	1071496490	1071562026	0.996065759
96	1071627562	11111111011111101111010010101010	1071496490	1071562026	0.996065759
97	1071627562	11111111011111101111010010101010	1071496490	1071562026	0.996065759
98	1071627562	11111111011111101111010010101010	1071496490	1071588240	0.996065759
99	1071627562	11111111011111101111010010101010	1054850346	1069910519	0.996065759

100	1071627562	11111111011111101111010010101010	1054850346	1069925264	0.996065759

El mejor valor de x entre todas las generaciones fue 1071627562, y su equivalente en binario es 11111111011111101111010010101010.

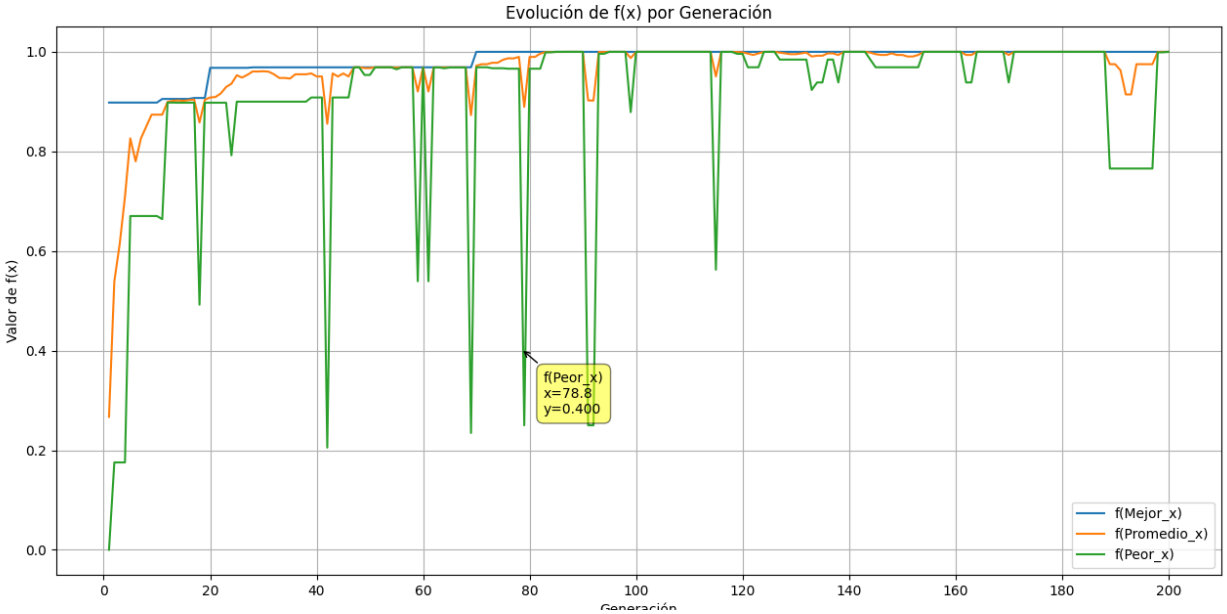

Figura 9: 200 generaciones - Método de Elitismo

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	1017527227	11110010100110001110111011011	7562754	554922747	0.898033089
2	1017527227	11110010100110001110111011011	449783380	788384261.8	0.898033089
3	1017527227	11110010100110001110111011011	449783329	840870753.7	0.898033089
4	1017527227	11110010100110001110111011011	449783329	904223092	0.898033089
5	1017527227	11110010100110001110111011011	879037217	975980223	0.898033089
6	1017527227	11110010100110001110111011011	879037217	948297815.6	0.898033089
7	1017527227	11110010100110001110111011011	879037361	975988017.8	0.898033089
8	1017527227	11110010100110001110111011011	879037361	989829205.6	0.898033089
9	1017527227	11110010100110001110111011011	879037361	1003652022	0.898033089
10	1017527227	11110010100110001110111011011	879037361	1003652021	0.898033089
11	1021643697	11110011100101000010111011000 1	874920881	1003678235	0.905313884
12	1021655985	11110011100101001110111011000 1	1017514929	1018350519	0.905335662
13	1021655985	11110011100101001110111011000 1	1017514929	1019586688	0.905335662
14	1021655985	11110011100101001110111011000 1	1017383857	1019959013	0.905335662
15	1021709233	11110011100110000010111011000 1	1017383857	1019531377	0.905430035
16	1021709233	11110011100110000010111011000 1	1017383857	1020342385	0.905430035
17	1022757809	11110011110110000010111011000 1	1017383857	1020904753	0.907289467
18	1022757809	11110011110110000010111011000 1	753273649	994565028.2	0.907289467
19	1022757809	11110011110110000010111011000 1	1017383857	1020654923	0.907289467

20	1056312241	11111011110110000010111011000 1	1017383857	1023321419	0.96779837
21	1056312241	11111011110110000010111011000 1	1017383857	1023858814	0.96779837
22	1056312241	11111011110110000010111011000 1	1017383857	1027856510	0.96779837
23	1056312241	11111011110110000010111011000 1	1017383857	1035104791	0.96779837
24	1056312241	11111011110110000010111011000 1	955648945	1038761700	0.96779837
25	1056312241	11111011110110000010111011000 1	1018563505	1048107134	0.96779837
26	1056312241	11111011110110000010111011000 1	1018563505	1045591371	0.96779837
27	1056312241	11111011110110000010111011000 1	1018563505	1048644529	0.96779837
28	1056836529	11111011111110000010111011000 1	1018563505	1052353873	0.968759319
29	1056852913	11111011111110010010111011000 1	1018563505	1052289969	0.968789356
30	1056852913	11111011111110010010111011000 1	1018563505	1052566961	0.968789356
31	1056852913	11111011111110010010111011000 1	1018563505	1052073700	0.968789356
32	1056852913	11111011111110010010111011000 1	1018563505	1049136049	0.968789356
33	1056852913	11111011111110010010111011000 1	1018563505	1045151460	0.968789356
34	1056852913	11111011111110010010111011000 1	1018563505	1045154743	0.968789356
35	1056852913	11111011111110010010111011000 1	1018563505	1044473163	0.968789356
36	1056852921	11111011111110010010111011001 1	1018563505	1049190117	0.968789371
37	1056852921	11111011111110010010111011001 1	1018563505	1049191755	0.968789371
38	1056852921	11111011111110010010111011001 1	1018563505	1049195033	0.968789371
39	1056852921	11111011111110010010111011001 1	1023298489	1050142030	0.968789371
40	1056852921	11111011111110010010111011001 1	1023298489	1046786589	0.968789371
41	1056852921	11111011111110010010111011001 1	1023298361	1046786579	0.968789371
42	1056852921	11111011111110010010111011001 1	486427577	993099498.6	0.968789371
43	1056852921	11111011111110010010111011001 1	1023298489	1050142035	0.968789371
44	1056852921	11111011111110010010111011001 1	1023298489	1046786591	0.968789371
45	1056852921	11111011111110010010111011001 1	1023298489	1050142854	0.968789371
46	1056852921	11111011111110010010111011001 1	1023298489	1046786591	0.968789371
47	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
48	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
49	1056852921	11111011111110010010111011001 1	1048464313	1056014060	0.968789371
50	1056852921	11111011111110010010111011001 1	1048464313	1056014060	0.968789371
51	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
52	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
53	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
54	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
55	1056852921	11111011111110010010111011001 1	1054755769	1056643206	0.968789371
56	1056852921	11111011111110010010111011001 1	1056852921	1056852921	0.968789371
57	1056918457	11111011111111010010111011001 1	1056852905	1056859473	0.968909525
58	1056918457	11111011111111010010111011001 1	1056852905	1056866025	0.968909525
59	1056918457	11111011111111010010111011001 1	788417465	1030035588	0.968909525
60	1056918457	11111011111111010010111011001 1	1056852905	1056885684	0.968909525
61	1056918457	11111011111111010010111011001 1	788483001	1030029035	0.968909525
62	1056918457	11111011111111010010111011001 1	1056656313	1056833257	0.968909525

63	1056918457	11111011111111010010111011100 1	1056656313	1056787385	0.968909525
64	1056918457	11111011111111010010111011100 1	1055869881	1056734956	0.968909525
65	1056918457	11111011111111010010111011100 1	1056656313	1056839814	0.968909525
66	1056918457	11111011111111010010111011100 1	1056656313	1056839814	0.968909525
67	1056918457	11111011111111010010111011100 1	1056656313	1056813599	0.968909525
68	1056918457	11111011111111010010111011100 1	1056656305	1056813599	0.968909525
69	1056918457	11111011111111010010111011100 1	520047545	1003178937	0.968909525
70	1073695673	11111111111111010010111011100 1	1056918457	1058596179	0.999914041
71	1073695673	11111111111111010010111011100 1	1056918457	1060273900	0.999914041
72	1073695673	11111111111111010010111011100 1	1056918457	1060273900	0.999914041
73	1073695673	11111111111111010010111011100 1	1055869881	1061846764	0.999914041
74	1073695673	11111111111111010010111011100 1	1055869881	1061846763	0.999914041
75	1073695673	11111111111111010010111011100 1	1055869881	1065097350	0.999914041
76	1073695673	11111111111111010010111011100 1	1055345593	1066722643	0.999914041
77	1073695673	11111111111111010010111011100 1	1055345593	1066565355	0.999914041
78	1073695673	11111111111111010010111011100 1	1055345593	1068243077	0.999914041
79	1073695673	11111111111111010010111011100 1	536824761	1012668550	0.999914041
80	1073695673	11111111111111010010111011100 1	1055345593	1068190649	0.999914041
81	1073695673	11111111111111010010111011100 1	1055345593	1068190649	0.999914041
82	1073695673	11111111111111010010111011100 1	1055345593	1071703379	0.999914041
83	1073695673	11111111111111010010111011100 1	1073171385	1073538387	0.999914041
84	1073695673	11111111111111010010111011100 1	1073171385	1073590815	0.999914041
85	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
86	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
87	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
88	1073695673	11111111111111010010111011100 1	1073695417	1073695647	0.999914041
89	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
90	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
91	1073695673	11111111111111010010111011100 1	536824761	1020008582	0.999914041
92	1073695673	11111111111111010010111011100 1	536824761	1019798867	0.999914041
93	1073695673	11111111111111010010111011100 1	1071598521	1073485958	0.999914041
94	1073695673	11111111111111010010111011100 1	1071598521	1073485958	0.999914041
95	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
96	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
97	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
98	1073695673	11111111111111010010111011100 1	1073695673	1073695673	0.999914041
99	1073695673	11111111111111010010111011100 1	1006586809	1066984787	0.999914041
100	1073695675	11111111111111010010111011101 1	1073695673	1073695673	0.999914044
101	1073695675	11111111111111010010111011101 1	1073695673	1073695673	0.999914044
102	1073695675	11111111111111010010111011101 1	1073695673	1073695673	0.999914044
103	1073695675	11111111111111010010111011101 1	1073695673	1073695674	0.999914044
104	1073695675	11111111111111010010111011101 1	1073695673	1073695675	0.999914044
105	1073695675	11111111111111010010111011101 1	1073695673	1073695675	0.999914044

106	1073695675	11111111111111010010111011011	1073695673	1073695675	0.999914044
107	1073695675	11111111111111010010111011011	1073695659	1073695673	0.999914044
108	1073695675	11111111111111010010111011011	1073695659	1073695672	0.999914044
109	1073695675	11111111111111010010111011011	1073695659	1073695673	0.999914044
110	1073695675	11111111111111010010111011011	1073695659	1073695672	0.999914044
111	1073695675	11111111111111010010111011011	1073695659	1073695670	0.999914044
112	1073695675	11111111111111010010111011011	1073695659	1073695670	0.999914044
113	1073695675	11111111111111010010111011011	1073695659	1073695672	0.999914044
114	1073695675	11111111111111010010111011011	1073695659	1073695670	0.999914044
115	1073695675	11111111111111010010111011011	805260219	1046852126	0.999914044
116	1073695675	11111111111111010010111011011	1073630139	1073689120	0.999914044
117	1073695675	11111111111111010010111011011	1073630139	1073682566	0.999914044
118	1073695675	11111111111111010010111011011	1073630139	1073682565	0.999914044
119	1073695675	11111111111111010010111011011	1071532987	1073472824	0.999914044
120	1073695675	11111111111111010010111011011	1071532987	1073256581	0.999914044
121	1073695675	11111111111111010010111011011	1056918459	1071782024	0.999914044
122	1073695675	11111111111111010010111011011	1056918459	1070327125	0.999914044
123	1073695675	11111111111111010010111011011	1056918459	1072011400	0.999914044
124	1073695675	11111111111111010010111011011	1073564603	1073682568	0.999914044
125	1073695675	11111111111111010010111011011	1073695675	1073695675	0.999914044
126	1073695675	11111111111111010010111011011	1073695675	1073695675	0.999914044
127	1073695675	11111111111111010010111011011	1065307067	1072856814	0.999914044
128	1073695675	11111111111111010010111011011	1065307067	1072017953	0.999914044
129	1073695675	11111111111111010010111011011	1065307067	1071179092	0.999914044
130	1073695675	11111111111111010010111011011	1065307067	1071179093	0.999914044
131	1073695675	11111111111111010010111011011	1065307067	1072017953	0.999914044
132	1073695675	11111111111111010010111011011	1065307067	1072856814	0.999914044
133	1073695675	11111111111111010010111011011	1031752635	1068662510	0.999914044
134	1073703867	11111111111111011010111011011	1040141243	1069502190	0.999929303
135	1073703867	11111111111111011010111011011	1040141243	1069502971	0.999929303
136	1073703867	11111111111111011010111011011	1065315259	1072021230	0.999929303
137	1073703867	11111111111111011010111011011	1065315259	1072020411	0.999929303
138	1073703867	11111111111111011010111011011	1040141243	1070341870	0.999929303
139	1073703867	11111111111111011010111011011	1073695675	1073703048	0.999929303
140	1073703931	11111111111111011010111111011	1073701819	1073703669	0.999929422
141	1073703931	11111111111111011010111111011	1073701819	1073703681	0.999929422
142	1073703931	11111111111111011010111111011	1073701819	1073703483	0.999929422
143	1073703931	11111111111111011010111111011	1073701819	1073703483	0.999929422
144	1073703931	11111111111111011010111111011	1065315323	1072864437	0.999929422
145	1073703931	11111111111111011010111111011	1056924667	1071186523	0.999929422
146	1073703931	11111111111111011010111111011	1056924667	1070347873	0.999929422
147	1073703931	11111111111111011010111111011	1056924667	1070347873	0.999929422
148	1073703931	11111111111111011010111111011	1056926715	1072025800	0.999929422

149	1073703931	11111111111111011010111111111011	1056926203	1070348229	0.999929422
150	1073703931	11111111111111011010111111111011	1056926715	1070348027	0.999929422
151	1073703931	11111111111111011010111111111011	1056926715	1068670715	0.999929422
152	1073703931	11111111111111011010111111111011	1056926715	1068669947	0.999929422
153	1073703931	11111111111111011010111111111011	1056926715	1070347617	0.999929422
154	1073703931	11111111111111011010111111111011	1073703419	1073703880	0.999929422
155	1073703931	11111111111111011010111111111011	1073703419	1073703777	0.999929422
156	1073703931	11111111111111011010111111111011	1073703419	1073703879	0.999929422
157	1073703931	11111111111111011010111111111011	1073703419	1073703880	0.999929422
158	1073703931	11111111111111011010111111111011	1073703419	1073703880	0.999929422
159	1073703931	11111111111111011010111111111011	1073703419	1073703880	0.999929422
160	1073703931	11111111111111011010111111111011	1073703419	1073703880	0.999929422
161	1073703931	11111111111111011010111111111011	1073703803	1073703918	0.999929422
162	1073703931	11111111111111011010111111111011	1040149499	1070348488	0.999929422
163	1073703931	11111111111111011010111111111011	1040149499	1070348488	0.999929422
164	1073703931	11111111111111011010111111111011	1073703931	1073703931	0.999929422
165	1073703931	11111111111111011010111111111011	1073703899	1073703928	0.999929422
166	1073703931	11111111111111011010111111111011	1073703931	1073703931	0.999929422
167	1073703935	11111111111111011010111111111111	1073703931	1073703931	0.999929429
168	1073703935	11111111111111011010111111111111	1073703931	1073703932	0.999929429
169	1073703935	11111111111111011010111111111111	1073703931	1073703932	0.999929429
170	1073703935	11111111111111011010111111111111	1040149503	1070348490	0.999929429
171	1073703935	11111111111111011010111111111111	1073703931	1073703933	0.999929429
172	1073703935	11111111111111011010111111111111	1073703931	1073703933	0.999929429
173	1073703935	11111111111111011010111111111111	1073703931	1073703933	0.999929429
174	1073703935	11111111111111011010111111111111	1073703931	1073703933	0.999929429
175	1073703935	11111111111111011010111111111111	1073703931	1073703933	0.999929429
176	1073703935	11111111111111011010111111111111	1073703930	1073703933	0.999929429
177	1073703935	11111111111111011010111111111111	1073703930	1073703934	0.999929429
178	1073703935	11111111111111011010111111111111	1073703930	1073703933	0.999929429
179	1073703935	11111111111111011010111111111111	1073703930	1073703933	0.999929429
180	1073703935	11111111111111011010111111111111	1073703930	1073703932	0.999929429
181	1073703935	11111111111111011010111111111111	1073703930	1073703934	0.999929429
182	1073703935	11111111111111011010111111111111	1073703930	1073703934	0.999929429
183	1073703935	11111111111111011010111111111111	1073703930	1073703934	0.999929429
184	1073703935	11111111111111011010111111111111	1073703930	1073703934	0.999929429
185	1073703935	11111111111111011010111111111111	1073703935	1073703935	0.999929429
186	1073703935	11111111111111011010111111111111	1073703935	1073703935	0.999929429
187	1073703935	11111111111111011010111111111111	1073703933	1073703935	0.999929429
188	1073703935	11111111111111011010111111111111	1073703935	1073703935	0.999929429
189	1073703935	11111111111111011010111111111111	939486207	1060282162	0.999929429
190	1073703935	11111111111111011010111111111111	939486207	1060282162	0.999929429
191	1073703935	11111111111111011010111111111111	939486207	1053571275	0.999929429

192	1073703935	11111111111111011010111111111	939486207	1026727730	0.999929429
193	1073703935	11111111111111011010111111111	939486207	1026727730	0.999929429
194	1073703935	11111111111111011010111111111	939486205	1060282161	0.999929429
195	1073703935	11111111111111011010111111111	939486205	1060282162	0.999929429
196	1073703935	11111111111111011010111111111	939486207	1060282162	0.999929429
197	1073703935	11111111111111011010111111111	939486207	1060282162	0.999929429
198	1073703935	11111111111111011010111111111	1073179647	1073651506	0.999929429
199	1073703935	11111111111111011010111111111	1073179647	1073651506	0.999929429
200	1073703935	11111111111111011010111111111	1073703935	1073703935	0.999929429

El mejor valor de x entre todas las generaciones fue 1073703935, y su equivalente en binario es 11111111111111011010111111111.

6.4. Modificaciones

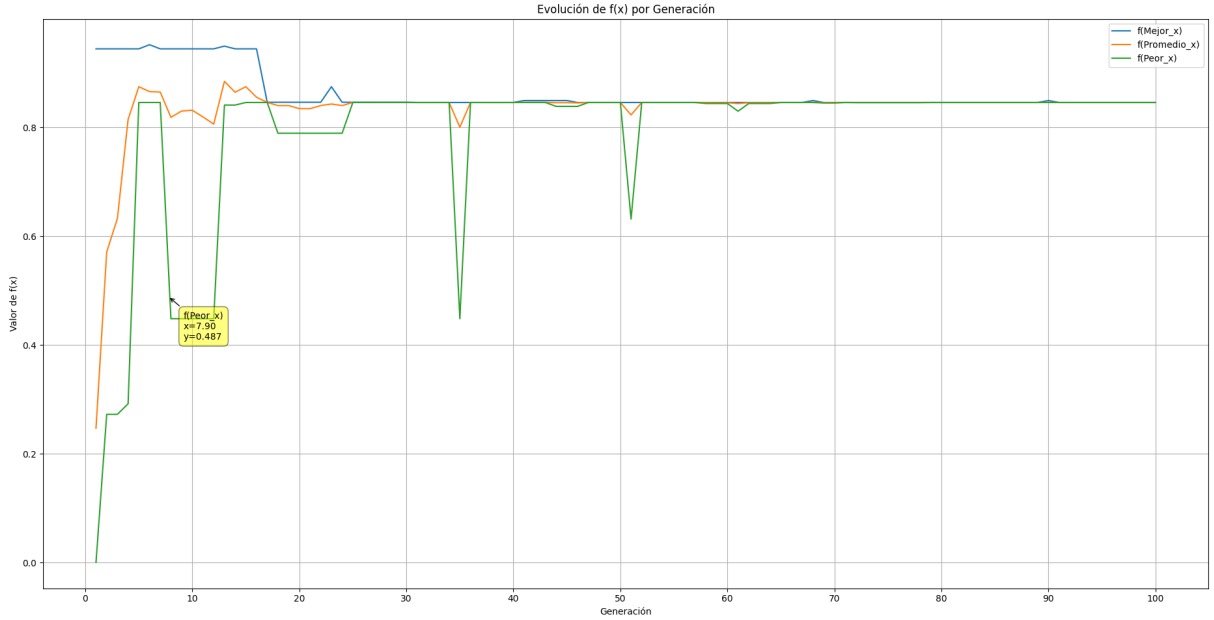

Figura 10: 100 generaciones - Método de Ruleta con PC = 0.1 y PM = 0.05

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	1043351026	11111000110000010001011111 0010	22989598	533729508.4	0.944193825
2	1043351026	11111000110000010001011111 0010	560587285	811374212	0.944193825
3	1043351026	11111000110000010001011111 0010	560587285	854055051.9	0.944193825
4	1043351026	11111000110000010001011111 0010	580206066	969058074.8	0.944193825
5	1043351027	11111000110000010001011111 0011	987394121	1004181346	0.944193827
6	1047545330	11111001110000010001011111 0010	987394376	999005137.1	0.951800462
7	1043351026	11111000110000010001011111 0010	987394377	998585706.8	0.944193825
8	1043351026	11111000110000010001011111 0010	718958921	971322730.8	0.944193825

9	1043351026	11111000110000010001011111 0010	718958921	978176686.9	0.944193825
10	1043351026	11111000110000010001011111 0010	718958921	979015547.7	0.944193825
11	1043351026	11111000110000010001011111 0010	718958921	971742161.2	0.944193825
12	1043351026	11111000110000010001011111 0010	718958921	963916056.5	0.944193825
13	1046114633	11111001011010011100010100 1001	984630770	1009777037	0.949202372
14	1043351026	11111000110000010001011111 0010	984630770	998309346.1	0.944193825
15	1043351026	11111000110000010001011111 0010	987394377	1004207586	0.944193825
16	1043351026	11111000110000010001011111 0010	987394377	993016256.3	0.944193825
17	987656521	11101011011110011100010100 1001	987394377	987473020.2	0.846081369
18	987656513	11101011011110011100010100 0001	953839945	984071701	0.846081356
19	987656513	11101011011110011100010100 0001	953839945	984104468.2	0.846081356
20	987656513	11101011011110011100010100 0001	953839945	980755578.6	0.846081356
21	987656513	11101011011110011100010100 0001	953839945	980742471.4	0.846081356
22	987656513	11101011011110011100010100 0001	953839945	984097914.6	0.846081356
23	1004171593	11101111011010011100010100 1001	953839945	985769082.6	0.874613394
24	987656513	11101011011110011100010100 0001	953839977	984091364.2	0.846081356
25	987656513	11101011011110011100010100 0001	987394377	987446804.2	0.846081356
26	987656513	11101011011110011100010100 0001	987394377	987446804.2	0.846081356
27	987656513	11101011011110011100010100 0001	987394377	987445985	0.846081356
28	987648321	11101011011110010100010100 0001	987394377	987419771.4	0.84606732
29	987648321	11101011011110010100010100 0001	987394377	987419771.4	0.84606732
30	987648321	11101011011110010100010100 0001	987394377	987419873.8	0.84606732
31	987395401	11101011011010011101010100 1001	987387209	987393865	0.845634049
32	987394377	11101011011010011100010100 1001	987387209	987392943.4	0.845632295
33	987394377	11101011011010011100010100 1001	987394377	987394377	0.845632295
34	987394377	11101011011010011100010100 1001	987394377	987394377	0.845632295
35	987394377	11101011011010011100010100 1001	718958921	960550831.4	0.845632295
36	987394377	11101011011010011100010100 1001	987394377	987394377	0.845632295
37	987394377	11101011011010011100010100 1001	987377993	987392738.6	0.845632295
38	987394377	11101011011010011100010100 1001	987377993	987392713	0.845632295
39	987394377	11101011011010011100010100 1001	987377993	987392687.4	0.845632295
40	987394377	11101011011010011100010100 1001	987394121	987394274.6	0.845632295
41	989491529	11101011111010011100010100 1001	987394121	987603938.6	0.849228229
42	989491529	11101011111010011100010100 1001	987394121	987603913	0.849228229
43	989491529	11101011111010011100010100 1001	987394121	987603938.6	0.849228229
44	989491529	11101011111010011100010100 1001	983199817	987184508.2	0.849228229
45	989491529	11101011111010011100010100 1001	983199817	987184610.6	0.849228229
46	987394633	11101011011010011100100100 1001	983199817	986974895.4	0.845632733
47	987394633	11101011011010011100100100 1001	987394121	987394377.2	0.845632733
48	987394633	11101011011010011100100100 1001	987394121	987394326.2	0.845632733
49	987394633	11101011011010011100100100 1001	987394121	987394224	0.845632733
50	987394633	11101011011010011100100100 1001	987394121	987394275.2	0.845632733
51	987394633	11101011011010011100100100 1001	853176395	973972502.8	0.845632733

52	987394633	11101011011010011100100100 1001	987394121	987394428.8	0.845632733
53	987394633	11101011011010011100100100 1001	987394059	987394422.2	0.845632733
54	987394633	11101011011010011100100100 1001	987394059	987394364.8	0.845632733
55	987394633	11101011011010011100100100 1001	987394059	987394217.8	0.845632733
56	987394633	11101011011010011100100100 1001	987394059	987394166.6	0.845632733
57	987394633	11101011011010011100100100 1001	987394059	987394166.4	0.845632733
58	987394633	11101011011010011100100100 1001	986345545	987289335	0.845632733
59	987394377	11101011011010011100010100 1001	986345545	987184433	0.845632295
60	987394377	11101011011010011100010100 1001	986345545	987184457.8	0.845632295
61	987394379	11101011011010011100010100 1011	977956937	986240790.4	0.845632298
62	987394379	11101011011010011100010100 1011	986345545	987184534.2	0.845632298
63	987394377	11101011011010011100010100 1001	986345545	987184533.8	0.845632295
64	987394377	11101011011010011100010100 1001	986345801	987183765.8	0.845632295
65	987394377	11101011011010011100010100 1001	987394121	987394223.4	0.845632295
66	987394377	11101011011010011100010100 1001	987394121	987394150.2	0.845632295
67	987394153	11101011011010011100000110 1001	987394121	987394127.8	0.845631911
68	989491273	11101011111010011100000100 1001	987394120	987603839.3	0.84922779
69	987394153	11101011011010011100000110 1001	986869833	987341701.8	0.845631911
70	987394153	11101011011010011100000110 1001	986869833	987341705	0.845631911
71	987394153	11101011011010011100000110 1001	987394121	987394137	0.845631911
72	987394153	11101011011010011100000110 1001	987263081	987381026.6	0.845631911
73	987394153	11101011011010011100000110 1001	987263081	987381026.6	0.845631911
74	987394153	11101011011010011100000110 1001	987263081	987381023.8	0.845631911
75	987394153	11101011011010011100000110 1001	987263081	987381020.2	0.845631911
76	987394153	11101011011010011100000110 1001	987263081	987381020.2	0.845631911
77	987394121	11101011011010011100000100 1001	987263081	987381017	0.845631856
78	987394121	11101011011010011100000100 1001	987263081	987381017	0.845631856
79	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
80	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
81	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
82	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
83	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
84	987394121	11101011011010011100000100 1001	987394113	987394120.2	0.845631856
85	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
86	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
87	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
88	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
89	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
90	989491273	11101011111010011100000100 1001	987394121	987603836.2	0.84922779
91	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
92	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856
93	987394125	11101011011010011100000100 1101	987394121	987394121.4	0.845631863
94	987394121	11101011011010011100000100 1001	987394121	987394121	0.845631856

95	987394121	11101011011010011100000 1001001	987394121	987394121	0.845631856
96	987394121	11101011011010011100000 1001001	987394121	987394121	0.845631856
97	987394121	11101011011010011100000 1001001	987394121	987394121	0.845631856
98	987394121	11101011011010011100000 1001001	987394121	987394121	0.845631856
99	987394153	11101011011010011100000 1101001	987394121	987394124.2	0.845631911
100	987396169	11101011011010011110000 1001001	987394121	987394325.8	0.845635364

El mejor valor de x entre todas las generaciones fue 1047545330, y su equivalente en binario es 11111001110000010001011111 0010.

En la figura anterior se observa la evolución a lo largo de 100 generaciones, utilizando selección por ruleta y una probabilidad de cruce significativamente reducida (de 0.75 a 0.1), pero manteniendo la probabilidad de mutación original. A diferencia de los casos anteriores, aquí el valor de la función objetivo del mejor individuo (curva azul) se mantiene casi constante durante buena parte del proceso evolutivo, evidenciando una falta de exploración del espacio de soluciones. No demuestra mejoras sostenidas ni acompañadas por un aumento progresivo del promedio poblacional.

La curva naranja, que representa el valor promedio de fitness de la población, se estabiliza rápidamente en niveles relativamente altos pero sin una tendencia de mejora continua. Esto refleja que, al haber pocas combinaciones genéticas entre individuos, el algoritmo tiende a conservar estructuras existentes sin generar nuevas soluciones de calidad. La baja probabilidad de cruce limita la recombinación entre individuos buenos.

Finalmente, la curva verde del peor individuo muestra picos descendentes que indican la presencia de mutaciones negativas, como en otros casos. En conjunto, la gráfica evidencia cómo una baja probabilidad de cruce compromete tanto la exploración como la explotación del algoritmo, generando un estancamiento temprano y limitando el rendimiento.

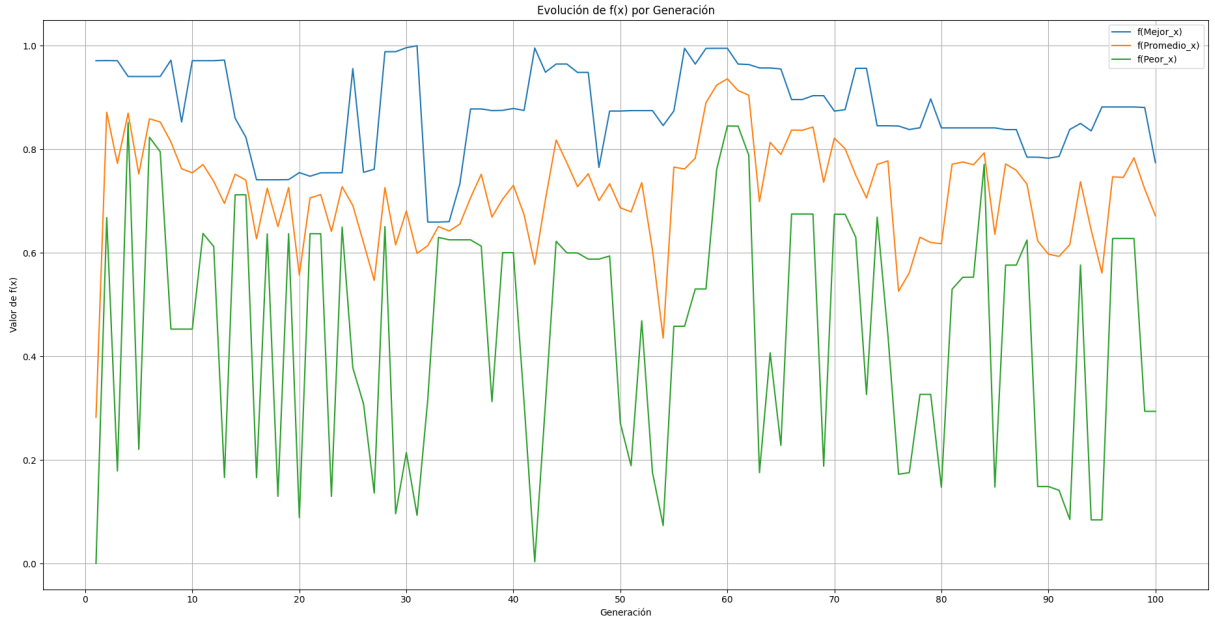

Figura 11: 100 generaciones - Método de Ruleta con PC = 0.75 y PM = 0.8

Generación	Mejor x	Mejor x (Bin)	Peor x	Promedio x	f(Mejor x)
1	1058021770	11111100010000001000011 0001010	21385694	570894540.3	0.970933462
2	1058152842	11111100010010001000011 0001010	877604318	1002326147	0.971174043
3	1058021762	11111100010000001000011 0000010	454041994	943832706.6	0.970933447

4	1041277759	11111000010000101000110011111	990913311	1001319787	0.940445095
5	1041277755	11111000010000101000110011011	504365450	931301404.1	0.940445087
6	1041277723	11111000010000101000110001101	974152074	995059629.3	0.94044503
7	1041343258	11111000010001101000110001101	957489983	991448447.3	0.940563411
8	1058562442	11111100011000011000011000101	722477886	968892781	0.971926052
9	991453578	11101100011000011000011000101	722477481	937489118.9	0.852599414
10	1058038185	11111100010000011000011010100	722477498	932616457.9	0.97096359
11	1058038189	11111100010000011000011010110	857227689	942489655.6	0.970963597
12	1058038186	11111100010000011000011010101	840100639	922423372.9	0.970963592
13	1058685353	11111100011010010000011010100	437840266	895408637.7	0.972151768
14	995681162	11101101011000111000111000101	906027392	930978503.3	0.859885928
15	974315946	11101000010010111000011010101	906027408	923898455.8	0.823379181
16	924328362	11011100011000001000011010101	437445034	850204471.7	0.741059056
17	924328378	11011100011000001000011011101	856711594	914129011.4	0.741059082
18	924328361	11011100011000001000011010100	387114376	866148263.4	0.741059055
19	924509352	11011100011010111001001010100	856860073	914956385.1	0.741349294
20	932897194	11011110011010111000011010101	319989161	801684987.7	0.754862471
21	928572904	11011101011000111001011110100	856973737	902173910.1	0.747880611
22	932636137	11011110010110111001011110100	856860073	906278218.9	0.754440057
23	932701674	11011110010111111001011110101	386966953	859827962.3	0.754546091
24	932701610	11011110010111111001011010101	865592810	915903070.1	0.754545987
25	1049658797	11111010010000100001011010110	660080072	892757446.3	0.955644931
26	933217706	11011110011111111001011010101	596264392	845089411	0.755381251
27	936895912	11011111010111111001011010100	396289450	793864220.6	0.76134754
28	1067380138	11111110011110111011010101010	866053514	914809575.6	0.988185541
29	1067445674	11111110011111111101101010101	333063658	842238676.6	0.988306892
30	1071572360	11111111011110111001011000100	496954666	885956681.9	0.995963143
31	1073669546	11111111111110111001011010101	328133048	831054974.1	0.999865378
32	871884216	11001111110111111001011011000	603448761	841145458.2	0.659352857
33	871884200	11001111110111111001011010100	852108713	866235014.1	0.659352832
34	872408488	11001111111111111001011010100	848962025	860399629.9	0.660146045
35	919217641	11011011001010001001011101001	848964009	869281879.6	0.732886905
36	1005986281	11101111110110001000011101001	848962920	902359385.7	0.877777364
37	1005987240	11101111110110001001011010100	840574368	930927656.4	0.877779038
38	1004186025	11101111011010101010011010100	600487401	878264252.4	0.874638534
39	1004446185	11101111011110101000011101001	831922665	900677352.3	0.875091788
40	1006543337	11101111111110101000011101001	831922537	917617179.5	0.878749756
41	1004315113	11101111011100101000011101001	600449441	881312876.2	0.874863418
42	1071292833	11111111011010101000011010001	63578529	816122669.3	0.995443603
43	1045832041	11111001010110001000010110100	596770920	900271751.6	0.948689616
44	1054515617	11111011011010101000011010001	847028589	971050861.6	0.964509018
45	1054517665	11111011011010101010011010001	831660393	944454087.3	0.964512765
46	1045602665	11111001010010101000010110100	831529257	915986370.4	0.948273521

47	1045635369	11111001010011001000010010 1001	823271720	931590143.2	0.948332842
48	939139561	11011111111010001000011110 1001	823271777	898807793.7	0.764998409
49	1003659753	11101111010010101000011110 1001	827466217	919538049.6	0.873722016
50	1003661289	11101111010010101001111110 1001	559030761	889975273.2	0.873724691
51	1004185577	11101111011010101001111110 1001	466788841	884827304.4	0.874637754
52	1004185577	11101111011010101001111110 1001	735225193	920860758	0.874637754
53	1004185579	11101111011010101001111110 1011	450536809	835513339.6	0.874637757
54	987408233	11101011011010101001110 1101001	291152041	708553514.6	0.845656028
55	1003758889	11101111010100001001010 0101001	726853480	939394312.9	0.873894628
56	1071023465	11111111010110100001010 1101001	726855529	937279714.9	0.994943073
57	1054483945	11111011011010001001011 1101001	781887337	949949613	0.964451082
58	1070867757	11111111010100001001010 0101101	781887337	1012926471	0.9946538
59	1070966249	11111111010101101001011 1101001	936650029	1032024932	0.994836773
60	1070966249	11111111010101101001011 1101001	986981677	1038743851	0.994836773
61	1054475325	11111011011010000001000 0111101	986850921	1026141786	0.964435314
62	1053992569	11111011010010101001100 1111001	953443913	1021128883	0.963552448
63	1050403653	11111010011011111000110 1000101	450127713	897626821.2	0.957001697
64	1050338892	11111010011010111001100 1001100	685007713	968255952.2	0.956883696
65	1049290314	11111010001010111001100 1001010	513044040	954311803.6	0.95497409
66	1016344140	11110010010100001011100 1001100	882140744	982366593.9	0.895946002
67	1016358472	11110010010100011001100 1001000	882124364	982106888.5	0.895971271
68	1020552777	11110011010100011001100 1001001	882124357	985795412.4	0.903381512
69	1020585545	11110011010100111001100 1001001	465749097	921332498.1	0.903439525
70	1003652201	11101111010010100001000 1101001	881862220	973154955.9	0.873708868
71	1005184585	11101111010011110011001 001001	881862220	961008305.9	0.876378876
72	1049953101	11111010010101000000 1101001101	852282189	929759157.7	0.956180895
73	1049985548	11111010010101100000 1000001100	613426793	902217654	0.956239994
74	987178572	11101011010111001001 1001001100	878192201	942572137	0.845262692
75	987211404	11101011010111101001 1010001100	714409476	946799980.9	0.845318917
76	986908196	11101011010011000001 1000100100	445973540	778605201.4	0.844799742
77	982984261	11101010010111001001 1001000101	450037284	804591137.3	0.83809527
78	984950595	11101010110101001001 1101000011	613754691	852304859.1	0.841451627
79	1017063233	11110010011111001001 1101000001	613754691	845386374.7	0.897214266
80	984819523	11101010110011001001 1101000011	412427843	843838566.8	0.841227691
81	984819011	11101010110011001001 0101000011	781526339	942828013.2	0.841226816
82	984819011	11101010110011001001 0101000011	798303555	945561386.2	0.841226816
83	984819206	11101010110011001001 1000000110	798451204	942225693.3	0.841227149
84	984821059	11101010110011001011 0101000011	942878019	956211666.7	0.841230315
85	984812867	11101010110011000011 0101000011	412706369	856178559.3	0.84121632
86	982903361	11101010010101111010 1001000001	815132262	943177194	0.837957324
87	982903380	11101010010101111010 1001010100	815132262	935614045.7	0.837957356
88	951152486	11100010110001011011 1101100110	848669249	918881926.3	0.784694404
89	951151889	11100010110001011011 0100010001	414281537	847532099.9	0.784693419

90	949936705	11100010011110111000100100001	414279232	829873960.6	0.782689664
91	952033857	11100010111110111000100100001	403901697	826948091.4	0.786149328
92	983026240	11101010010111110010100100000	313666630	842671821	0.838166854
93	989848740	11101011111111111100100101001 00	815194817	921964038.9	0.849841491
94	981371392	11101001111110100010100000000	311937760	861442313.2	0.835347252
95	1008192225	11110000010111110010101110000 1	311941735	804292137.3	0.881631196
96	1008175719	11110000010111100010100110011 1	850649252	927878696	0.881602328
97	1008175207	11110000010111100010000110011 1	850649254	927130684.4	0.881601432
98	1008175654	11110000010111100010100010011 0	850627144	950375529.5	0.881602214
99	1007610023	11110000001110111010001010011 1	582206567	913181143.6	0.880613256
100	944761455	11100001001111111010100110111 1	582206575	879921737	0.774184717

El mejor valor de x entre todas las generaciones fue 1073669546, y su equivalente en binario es 11111111111110111001011010101 0.

Para el experimento anterior se mantuvo la probabilidad de cruce original (0.75), pero se aumentó considerablemente la probabilidad de mutación de 0.05 a 0.8. Como resultado, el comportamiento de las curvas se vuelve mucho más errático, especialmente en el caso del peor individuo (curva verde), que presenta oscilaciones constantes y caídas abruptas durante prácticamente toda la evolución.

Este patrón refleja un exceso de ruido introducido por la mutación, que impide que la población se estabilice o conserve estructuras genéticas favorables. Aunque se logran ocasionales picos altos en el mejor individuo, estos no se mantienen en el tiempo, y la curva azul oscila fuertemente. El mecanismo de cruce queda opacado por la alta tasa de cambio que introduce la mutación. En conjunto, esta configuración conduce a una dinámica evolutiva caótica, donde el algoritmo tiene potencial para descubrir soluciones buenas, pero carece de estabilidad para preservarlas y hacerlas prevalecer.

7. Conclusiones

7.1. Método de Selección: Ruleta

A partir del análisis de las gráficas del método de selección por Ruleta, se observa que el algoritmo genético logra mejorar progresivamente - si bien de manera lenta - el valor del mejor individuo (mejor x) a lo largo de las generaciones. También se nota una suba en el promedio de la población, lo que indica que, en general, los cromosomas van mejorando generación tras generación. Esto demuestra que la Ruleta logra seleccionar con mayor probabilidad a los individuos con mejor fitness, y que el cruce permite generar nuevos individuos que potencian esa mejora.

Sin embargo, también se puede ver que hay algunas caídas o estancamientos temporales tanto en el promedio como en el peor individuo (peor x), lo que sugiere que a veces la población pierde diversidad genética, lo que coincide con la tabla de salidas. Por ejemplo, en casos de picos en bajada en el peor x, suele deberse a una mutación en un bit significativo. Aun así, en general, el método de Ruleta logra un buen equilibrio entre seleccionar a los mejores y mantener cierta variedad.

7.2. Método de Selección: Torneo

A medida que se incrementa la cantidad de generaciones, el algoritmo genético presenta una mejora progresiva en la calidad de las soluciones y a su vez mejora la eficiencia del mismo. En las corridas con 100 y 200 generaciones, se observa una rápida convergencia hacia valores altos de fitness, manteniéndose cerca del óptimo. La línea correspondiente al peor individuo presenta oscilaciones más notorias, lo cual indica que la población mantiene cierta diversidad genética, un aspecto positivo para evitar la convergencia prematura. La línea del promedio también se eleva progresivamente, reflejando una mejora general en el rendimiento de la población.

Es importante destacar que la línea del mejor individuo no presenta caídas, lo que sugiere que, aunque existe una probabilidad de mutación, el mejor individuo se mantiene a lo largo de las generaciones, ya sea porque no fue mutado o porque fue preservado mediante selección. En conjunto, el método de Torneo resulta eficaz, mostrando un comportamiento estable y eficiente a medida que aumentan las generaciones.

7.3. Método de Selección: Elitismo aplicado a ruleta

Los resultados obtenidos mostraron una rápida convergencia del valor máximo de la función objetivo hacia valores cercanos al óptimo, especialmente en las primeras generaciones, manteniéndose estable a lo largo del tiempo gracias al efecto del elitismo. Esta estrategia aseguró la retención de los mejores individuos, preservando el conocimiento evolutivo acumulado y garantizando la estabilidad de las soluciones óptimas en todas las configuraciones evaluadas (20, 100 y 200 generaciones).

Por otro lado, el comportamiento del peor individuo presentó fluctuaciones notables, manifestadas en picos descendentes pronunciados. Estas caídas se explican por la naturaleza probabilística de la selección por ruleta y la introducción de nuevos individuos poco aptos debido a las operaciones genéticas, especialmente la mutación. Sin embargo, estas oscilaciones no afectaron negativamente la convergencia general del algoritmo, evidenciando que el elitismo no elimina la diversidad, sino que permite un equilibrio entre la exploración y la explotación.

El valor promedio de la población mostró una evolución creciente y sostenida en todas las corridas, aunque con un ritmo más gradual que el mejor individuo. Este comportamiento indica que el elitismo no solo protege a los mejores, sino que también guía a la población hacia regiones de mayor aptitud, mejorando la calidad general del conjunto de soluciones.

Además, al aumentar el número de generaciones, el promedio de la población mejoró progresivamente y las curvas se estabilizaron, sugiriendo una convergencia efectiva del algoritmo. Incluso con un número reducido de generaciones (20), se lograron soluciones de alta calidad, aunque con mayor dispersión en el rendimiento.

En conjunto, la combinación de elitismo con selección por ruleta resultó ser una estrategia robusta para mantener un balance adecuado entre la explotación de soluciones óptimas y la exploración del espacio de búsqueda, favoreciendo tanto la estabilidad como la diversidad genética y, en consecuencia, la calidad y consistencia de los resultados obtenidos.

8. Conclusión General

Habiendo concluido la experiencia, notamos en líneas generales cómo evolucionó cada método de selección en base al método de selección y parámetros elegido, un aspecto clave al momento de evaluar e implementar un Algoritmo Genético.

En cuanto a los métodos de selección, se aprecia que la ruleta tiende a ser más aleatoria, lo que implica una mayor variabilidad en las primeras generaciones. Esto puede dificultar alcanzar soluciones óptimas en pocas iteraciones. Por otro lado, la selección por torneo introduce un control más firme sobre la calidad de los individuos elegidos, al comparar explícitamente sus niveles de fitness. Esto reduce la probabilidad de que un individuo muy malo sea seleccionado, aunque también puede dejar afuera a soluciones muy buenas si se topan con un torneo desfavorable, ya que el proceso tiene un componente aleatorio en la formación de los grupos.

Finalmente, la incorporación de elitismo demuestra ser clave para preservar el progreso del algoritmo. Al garantizar que los mejores individuos pasen a la siguiente generación sin modificaciones, se asegura que el nivel alcanzado no se pierda. Con suficientes iteraciones, este mecanismo permite consolidar el óptimo y mantenerlo. Las gráficas muestran cómo ciertos valores máximos se sostienen a lo largo de varias generaciones, lo que confirma que los cromosomas de mayor fitness se están manteniendo en la población. Esta práctica resulta especialmente útil cuando se quiere evitar que buenas soluciones desaparezcan por selección o mutación desfavorables.

En resumen, el análisis realizado permitió comprender en profundidad cómo distintos métodos de selección y configuraciones de parámetros afectan el comportamiento y la eficiencia de un Algoritmo Genético Canónico. Las ejecuciones múltiples y las gráficas obtenidas evidenciaron que factores como la probabilidad de cruce y mutación tienen un rol crucial en el equilibrio entre exploración y explotación. La comparación entre ruleta, torneo y elitismo mostró que no existe una única configuración ideal, sino que la efectividad del algoritmo depende del contexto y de la interacción entre los parámetros utilizados. En conjunto, los resultados obtenidos demuestran la importancia de ajustar cuidadosamente cada componente del algoritmo para garantizar una buena convergencia y la obtención de soluciones de alta calidad.

9. Bibliografía

[1] GeeksforGeeks. “Selection in Genetic Algorithms”. `https://www.geeksforgeeks.org/selection-in-genetic-algorithms/`.

[2] Matplotlib. “Using matplotlib.plyplot”. `https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html`.

[3] Python.org. “Documentation for Python”. `https://www.python.org/doc/`.

[4] Python Documentation. “Generate pseudo-random numbers”. https://docs.python.org/es/3.13/library/random.html.

[5] Towards Data Science. “Genetic Algorithms Explained: A Python Implementation”. https://towardsdatascience.com/genetic-algorithms-explained-a-python-implementation-6816e8c45ef4.

[6] Tutorialspoint. “Genetic Algorithm Basics”. https://www.tutorialspoint.com/genetic_algorithms/index.htm.