Memoria Práctica 2

Optimización de la Ruta de un Robot de Limpieza

21/03/2025

—

Programación Evolutiva

—

Bryan Xavier Quilumba Farinango

Introducción

Este proyecto implementa un algoritmo genético para optimizar la ruta de un robot de limpieza en una mansión modelada como una cuadrícula de 15x15. El objetivo esa recorrer 20 habitaciones en el orden determinado por la permutación del cromosoma, minimizando la distancia total recorrida. Se emplean operadores genéticos como selección, cruce y mutación para evolucionar soluciones más eficientes.

Esta es la segunda versión de la práctica con las correcciones, detalladas más adelante.

Resultados de las Ejecuciones

A continuación, se presentan los resultados de las cinco mejores ejecuciones (variando los operadores) del algoritmo (en esta versión con muchas menos generaciones necesarias). Pero antes una breve explicación del cruce, mutación y fitness.

Cruce Propio (Corte y Relleno Aleatorio)

Este cruce combina a dos padres para crear un hijo, mezclando un trozo de uno con genes aleatorios del otro.

Se copia un segmento aleatorio de un padre directamente al hijo. Y los huecos restantes en el hijo se rellenan con las habitaciones que faltan en un orden totalmente al azar.

Ejemplo: Si se copia el trozo [3, 4, 5] de un padre, el resto de posiciones [?, ?, ?, ?, ?] se llenan al azar con las habitaciones no usadas.

Mutación Propia (Mezcla de Segmento)

Esta mutación altera a un individuo barajando una pequeña parte de su ruta.

Se elige un segmento aleatorio de la ruta (por ejemplo, tres habitaciones seguidas). Y el orden de solo esas habitaciones se mezcla al azar. El resto de la ruta no cambia.

Ejemplo: Una ruta [1, 2, **3, 4, 5**, 6] podría mutar a [1, 2, **5, 3, 4**, 6] después de mezclar el trozo central.

Fitness

El fitness es simplemente la **distancia total** que recorre el robot para completar su ruta. Un valor de fitness más **bajo** significa que la ruta es más corta y, por lo tanto, **mejor**.
El proceso es el siguiente:

1. **Recorrido Completo:** El robot empieza en la BASE, visita cada una de las habitaciones en el orden que indica su cromosoma y, al terminar, regresa a la BASE.
2. **Cálculo de Tramos con A***: La distancia entre dos puntos (ej: de la habitación 3 a la 5) no es una línea recta, ya que hay obstáculos. Se utiliza el **algoritmo A*** (A estrella) para encontrar el camino más corto posible en el mapa para cada tramo del viaje.

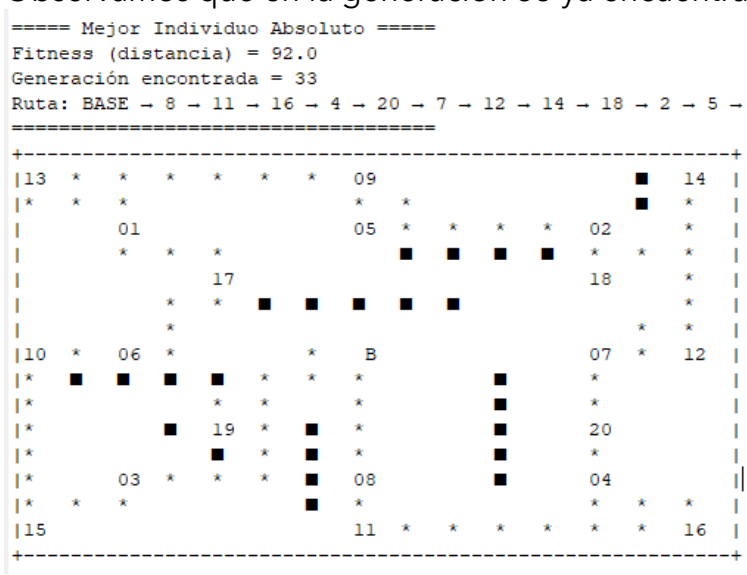
3. **Suma Total:** El fitness final es la **suma de las distancias** de todos esos tramos: (BASE → Hab. 1) + (Hab. 1 → Hab. 2) + ... + (Última Hab. → BASE).

Para que todo sea más rápido, el sistema usa una **caché (una "memoria")**: si ya ha calculado una ruta entre dos puntos, la guarda para usarla directamente la próxima vez sin tener que volver a calcularla.

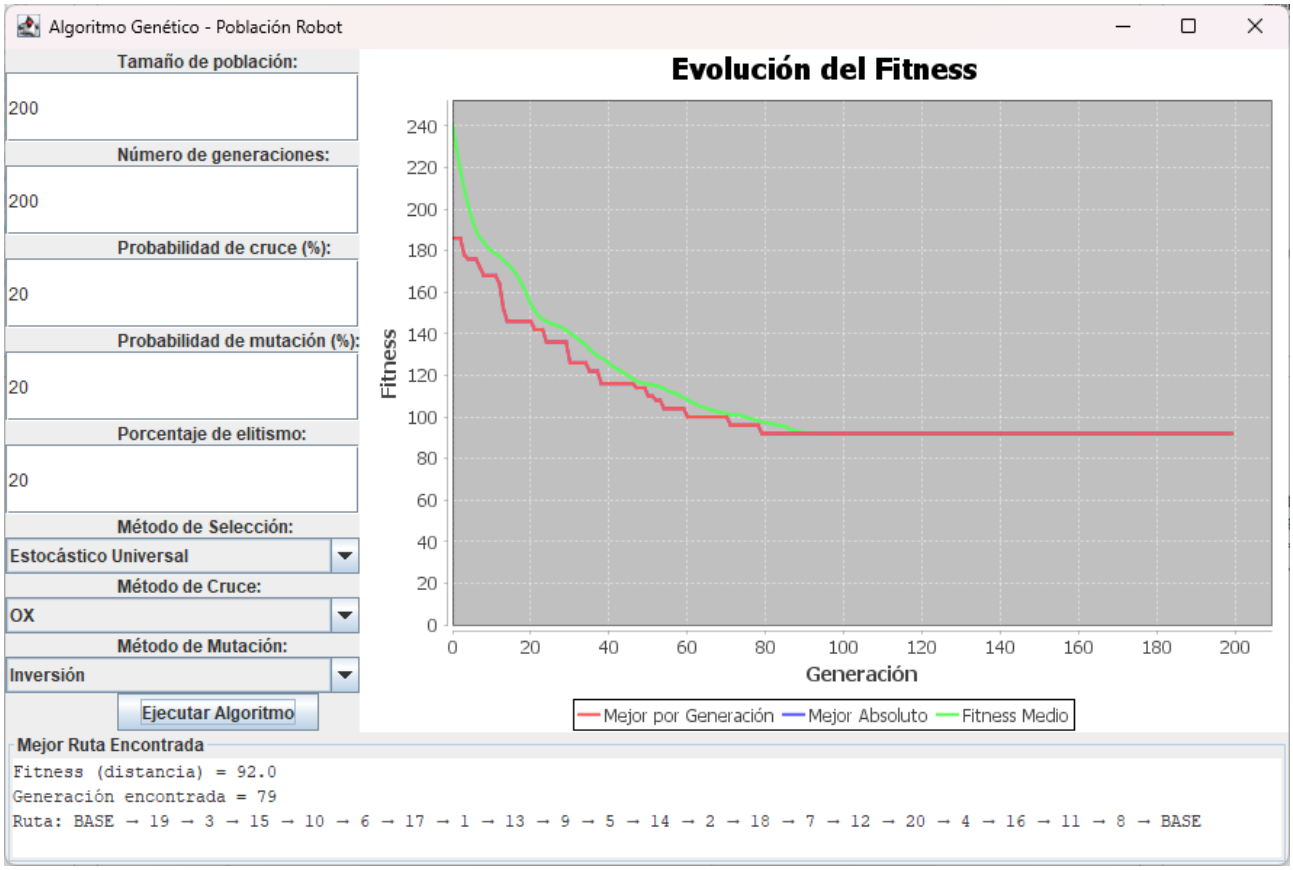
Ejecución 1

Observamos que en la generación 33 ya encuentra el óptimo. Mapa:

```
===== Mejor Individuo Absoluto =====
Fitness (distancia) = 92.0
Generación encontrada = 33
Ruta: BASE → 8 → 11 → 16 → 4 → 20 → 7 → 12 → 14 → 18 → 2 → 5 → 9 → 13 → 1 → 17 → 6 → 10 → 15 → 3 → 19 → BASE
====================================
```



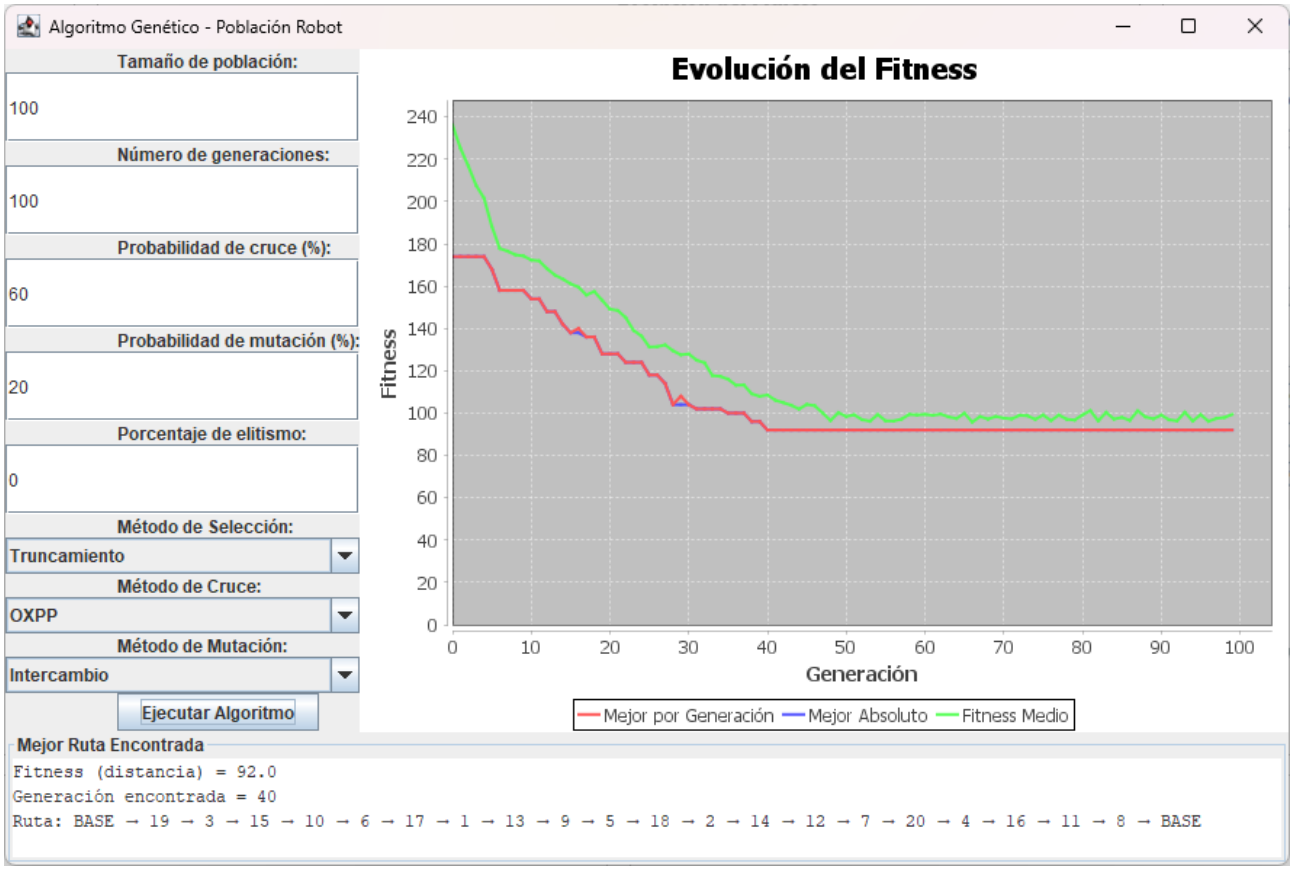
Ejecución 2



Con elitismo se encuentra una ejecución con el individuo óptimo de forma rápida.
Mapa:

```
===== Mejor Individuo Absoluto =====
Fitness (distancia) = 92.0
Generación encontrada = 79
Ruta: BASE → 19 → 3 → 15 → 10 → 6 → 17 → 1 → 13 → 9 → 5 → 14 → 2 → 18 → 7 → 12 → 20 → 4 → 16 → 11 → 8 → BASE
====================================
+------------------------------------------------------------+
|13  *   *   *   *   *   *   09                    ■   14   |
|*   *   *                   *   *                 ■   *    |
|         01                      05  *   *   *   *   02  *   *   |
|         *   *   *                   ■   ■   ■   ■   *   *       |
|                 17                                  18          |
|             *   *   ■   ■   ■   ■   ■               *           |
|     *   *   *                                       *           |
|10   *   06                       B                  07  *   12  |
|*    ■   ■   ■   ■   *   *   *               ■          *   *   |
|*        *   *   *   *       *               ■          *   *       |
|*        *   ■   19      ■   *               ■          20          |
|*        *       ■       ■   *               ■          *           |
|*        03              ■   08              ■          04          |
|*    *   *                   ■   *   *                  *   *   *   |
|15                               11  *   *   *   *   *   *   16  |
+------------------------------------------------------------+
```

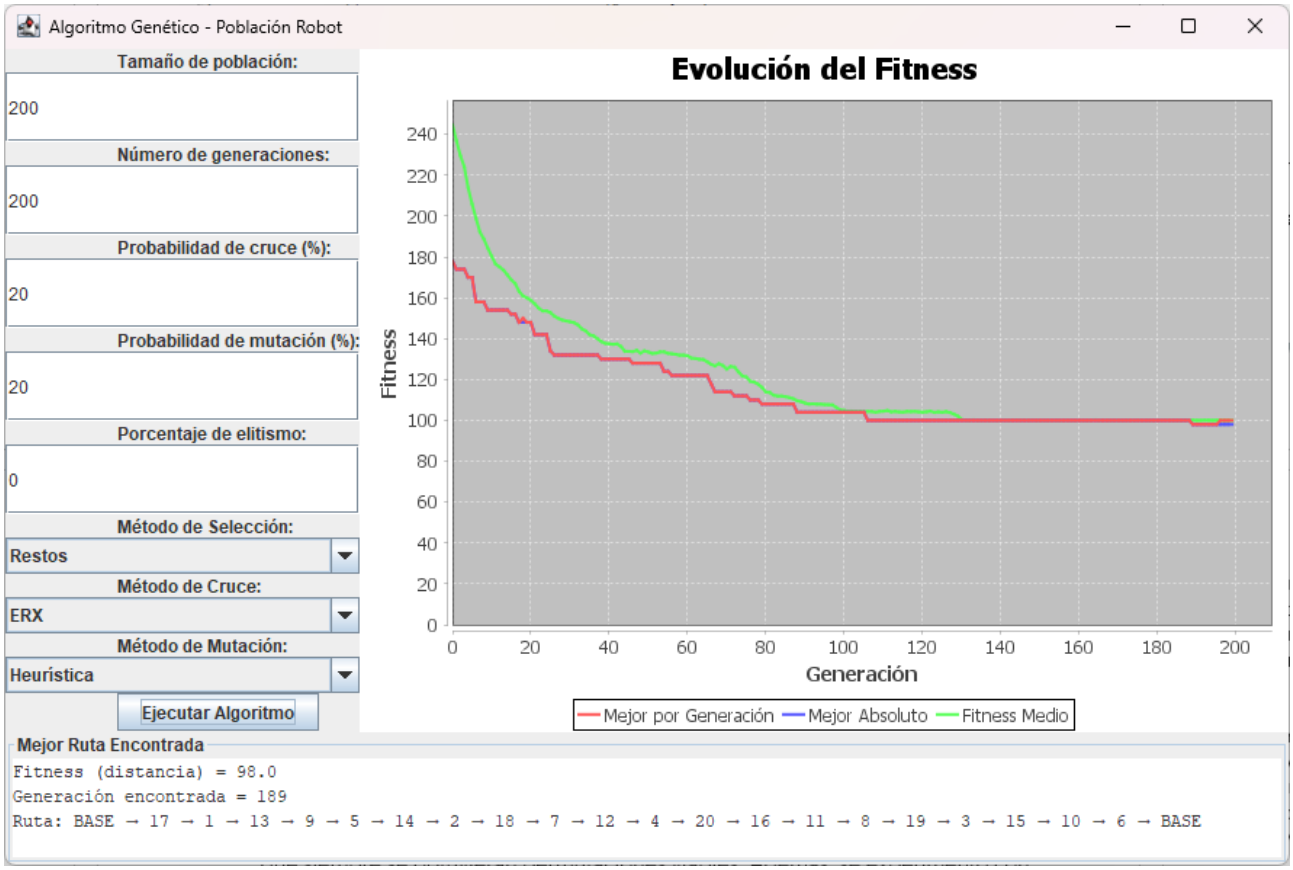
Ejecución 3



En esta ejecución he aumentado los cruces y llega al óptimo en la generación 40. Mapa:

```
===== Mejor Individuo Absoluto =====
Fitness (distancia) = 92.0
Generación encontrada = 40
Ruta: BASE → 19 → 3 → 15 → 10 → 6 → 17 → 1 → 13 → 9 → 5 → 18 → 2 → 14 → 12 → 7 → 20 → 4 → 16 → 11 → 8 → BASE
====================================
+------------------------------------------------------------+
|13  *   *   *   *   *   *   09                      ■   14  |
|*   *   *               *   *                       ■   *   |
|         01                  05  *   *   *   *   02  *   *   |
|         *   *   *               ■   ■   ■   ■   *       *   |
|                 17                                  18      *   |
|             *   *   ■   ■   ■   ■   ■                       *   |
|     *   *   *                                               *   |
|10  *   06                   B                       07  *   12  |
|*   ■   ■   ■   ■   *   *   *           ■           *   *       |
|*       *   *   *   *       *           ■           *           |
|*       *   ■   19      ■   *           ■           20          |
|*       *       ■       ■   *           ■           *           |
|*       03              ■   08          ■           04          |
|*   *   *               ■   *   *                   *   *   *   |
|15                           11  *   *   *   *   *   *   16  |
+------------------------------------------------------------+
```

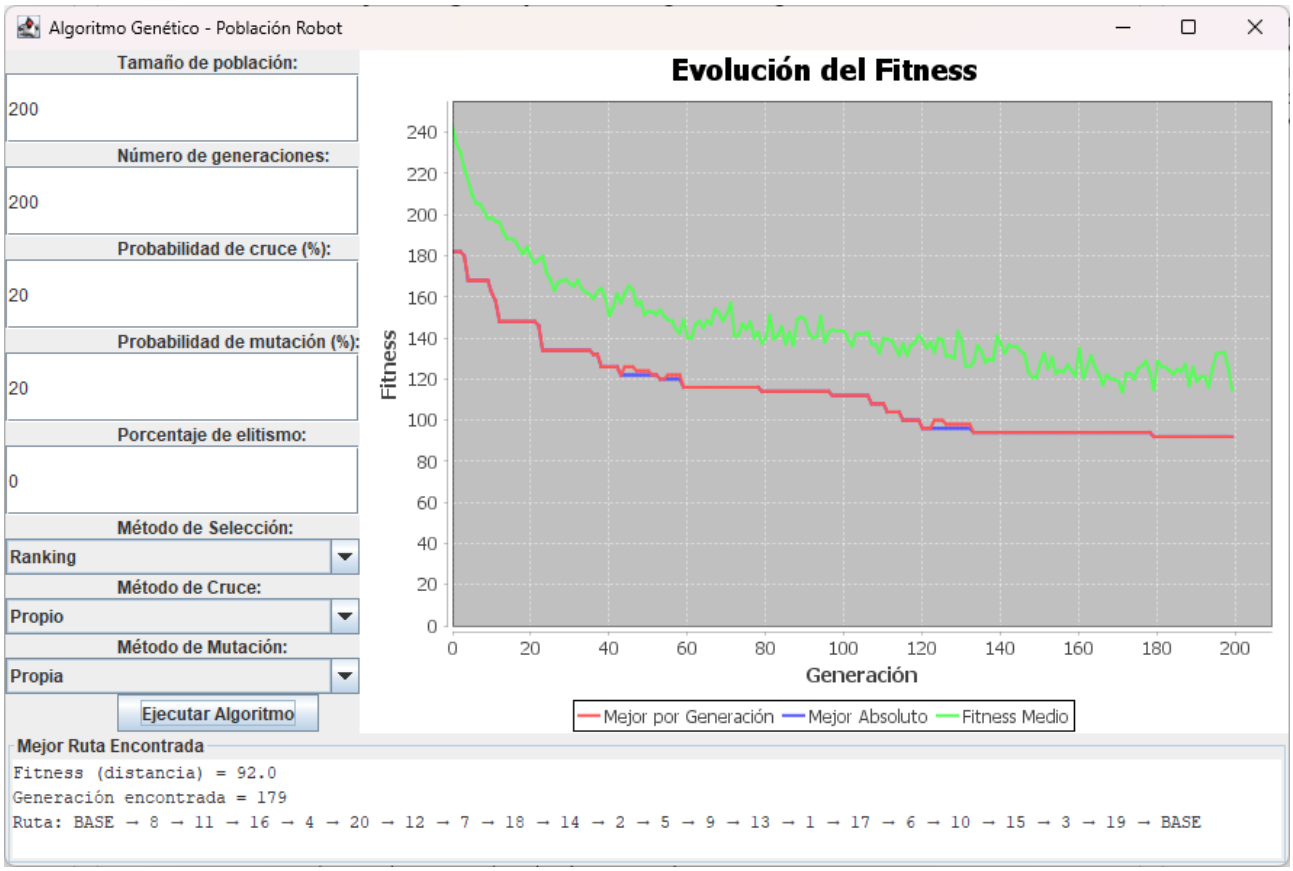
Ejecución 4



En esta ejecución no se llega al óptimo, pero se puede apreciar una evolución progresiva con la mutación heurística. Mapa:

```
===== Mejor Individuo Absoluto =====
Fitness (distancia) = 98.0
Generación encontrada = 189
Ruta: BASE → 17 → 1 → 13 → 9 → 5 → 14 → 2 → 18 → 7 → 12 → 4 → 20 → 16 → 11 → 8 → 19 → 3 → 15 → 10 → 6 → BASE
====================================
+------------------------------------------------------------+
|13  *   *   *   *   *   *   09                    ■   14   |
|*   *   *                   *   *                 ■   *    |
|         01                      05  *   *   *   *   02  *   *    |
|         *   *   *                   ■   ■   ■   ■   *   *        |
|                 17                                  18           |
|                 *   ■   ■   ■   ■   ■               *            |
|                 *   *   *   *                       *            |
|10  *   06  *   *   *   *    B                      07  *   12   |
|*   ■   ■   ■   ■                           ■           *   *    |
|*       *   *   *   *   *   *               ■               *    |
|*       *   ■   19  *   ■   *               ■       20      *    |
|*       *       ■       ■   *               ■       *   *   *    |
|*       03              ■   08              ■       04  *        |
|*   *   *                   ■   *   *                   *        |
|15                           11  *   *   *   *   *   *   16   |
+------------------------------------------------------------+
```

Ejecución 5



Y con unas pocas ejecuciones volvemos al individuo óptimo con operadores propios.
Mapa:

```
Fitness: 100.0
Ruta: 6→1→17→19→3→15→10→13→9→5→14→2→18→7→12→20→4→16→11→8
====================================
+-------------------------------------------+
|13 *  *  *  *  *  *  09              ■  14 |
|*                  *  *              ■  *  |
|*     01             05 *  *  *  *  02 *  * |
|*     *  *  *           ■  ■  ■  ■  *  *    |
|*     *     17                       18     |
|*     *     *  ■  ■  ■  ■  ■        *      |
|*     *  *  *                       *      |
|10    06 *  *  *  *   B             07 *  12 |
|*  ■  ■  ■  ■  *     *        ■        *  *  |
|*     *  *  *  *     *        ■     *  *     |
|*     *  ■  19 *  ■  *        ■     20       |
|*     *     ■     ■  *        ■     *        |
|*     03          ■  08       ■     04       |
|*  *  *           ■  *  *              *  *  * |
|15                   11 *  *  *  *  *  *  16 |
+-------------------------------------------+
```

Conclusiones

Durante la implementación del algoritmo genético, me he encontrado con varios desafíos que requirieron ajustes y mejoras:

- **Errores en la inicialización aleatoria de la población:** En las primeras versiones, algunos individuos tenían rutas repetidas o inválidas. Se solucionó asegurando que la generación inicial respetara restricciones de unicidad en las habitaciones visitadas.

- **Selección y convergencia prematura:** Algunas configuraciones de selección, como el torneo determinista, generaban convergencias tempranas a soluciones subóptimas. Se resolvió probando métodos como ruleta y ranking para mantener la diversidad de la población.
- **Cruce y mutación:** La implementación inicial de los operadores de cruce (PMX, OX, CX) generaba individuos inválidos. Ajustamos los métodos para asegurar que siempre se obtuvieran permutaciones viables. Además, se experimentó con diferentes tasas de mutación para evitar estancamientos.

- **Problemas de visualización:** La gráfica de evolución del fitness no se actualizaba correctamente debido a problemas en la revalidación del panel de gráficos. Se solucionó asegurando una actualización adecuada en la interfaz gráfica de Swing.

Estos problemas fueron resueltos mediante depuración, pruebas con distintos parámetros y ajustes en la lógica de los operadores genéticos, lo que permitió mejorar la eficiencia del algoritmo.

Descripción de la Implementación y Arquitectura

La aplicación se ha desarrollado utilizando las siguientes estructuras:

- **Algoritmo Genético:** Implementado en la clase AlgoritmoGenetico, encargada de la evolución de la población.
- **Población:** Representada por PoblacionRobot, gestiona la selección, cruce y mutación de individuos.
- **Individuo:** La clase IndividuoRobot modela la ruta de un robot y su evaluación.
- **Interfaz Gráfica:** Implementada en JFramePoblacionRobot, permite configurar los parámetros y visualizar los resultados.
- **Gráficos:** Utiliza la librería JFreeChart para mostrar la evolución del fitness.

El sistema está modularizado, lo que permite modificar fácilmente los operadores genéticos y la función de evaluación. Además, se han implementado diversos métodos de selección, cruce y mutación, lo que permite experimentar con diferentes configuraciones para mejorar la optimización de rutas.

Se han implementado mejoras en la representación de la población y en la tasa de mutación adaptativa, lo que ha permitido optimizar el rendimiento del algoritmo. El análisis de convergencia ha mostrado que, en la mayoría de los casos, el algoritmo encuentra soluciones óptimas en un número razonable de generaciones, aunque en problemas más complejos pueden observarse oscilaciones antes de estabilizarse. Durante la implementación, se ha seguido una arquitectura modular, facilitando la extensibilidad del código. La aplicación está organizada en módulos independientes para la inicialización, evaluación, selección, cruzamiento y mutación, permitiendo ajustes sencillos en cada componente.

Respecto, a las correcciones, principalmente se cambio la arquitectura, los operadores de cruce que generaban individuos erróneos, las mutaciones mal implementadas, y se explican los métodos de cruce y mutación propios, además del fitness.