

Modelos de Inteligencia Artificial

Profesor/a: Águeda María López Moreno

PRÁCTICA 1.4.- ENTENDIENDO LOS ALGORITMOS EVOLUTIVOS
21/10/2024

Índice

| | |
|--|-----------|
| 1.- Introducción | 3 |
| 1.1.- Enfoque del Algoritmo Genético para la Resolución del Problema | 3 |
| 1.2.- Objetivos del Análisis y Pruebas | 3 |
| 2.- Descripción del Código | 4 |
| 2.1.- Explicación de los Parámetros Principales | 4 |
| 2.2.- Estructura y Funciones Principales del Código | 5 |
| 3.- Modificaciones al Código Original | 6 |
| 3.1.- Ajustes en los Parámetros | 6 |
| 3.2.- Cambios en la Lógica de Selección | 7 |
| 3.3.- Mejora en el Proceso de Cruce | 7 |
| 3.4.- Ajuste en la Mutación | 7 |
| 3.5.- Análisis de Resultados y Registro de Tiempos | 8 |
| 4.- Pruebas Realizadas y Resultados Obtenidos | 8 |
| 4.1.- Configuraciones Probadas | 8 |
| 4.2.- Resultados de las Pruebas | 9 |
| 4.2.1.- Configuración 1: $n = 4$, $nPersonas = 10$, LIMITE = 20 | 9 |
| 4.2.2.- Configuración 2: $n = 10$, $nPersonas = 10$, LIMITE = 10 | 9 |
| 4.2.3.- Configuración 3: $n = 10$, $nPersonas = 20$, LIMITE = 5 | 10 |
| 4.2.4.- Configuración 4: $n = 10$, $nPersonas = 100$, LIMITE = 1 | 11 |
| 4.2.5.- Configuración 5: $n = 30$, $nPersonas = 100$, LIMITE = 0.5 | 11 |
| 4.3.- Análisis de Resultados | 12 |
| 5.- Análisis de Parámetros Óptimos para $n = 50$ | 12 |
| 5.1.- Selección de Parámetros Óptimos | 12 |
| 5.2.- Recomendación de Configuración | 13 |
| 5.3.- Justificación de la Configuración | 13 |
| 5.4.- Conclusión | 13 |
| 6.- Conclusión | 14 |
| 6.1.- Reflexión Final | 14 |
| 7.- Bibliografía | 15 |
| 8.- Esquema Resumido | 16 |

1.- Introducción

El problema de las N reinas es un clásico en la teoría de algoritmos y en la inteligencia artificial, que consiste en colocar N reinas en un tablero de ajedrez de tamaño $N \times N$ de manera que ninguna de ellas se ataque entre sí. Esto implica que no puede haber dos reinas en la misma fila, columna, o diagonal. Resolver este problema se vuelve cada vez más complejo a medida que aumenta el valor de N, debido al crecimiento exponencial en el número de combinaciones posibles.

1.1.- Enfoque del Algoritmo Genético para la Resolución del Problema

El algoritmo genético es una técnica inspirada en los procesos de selección natural y evolución biológica. Se utiliza ampliamente para problemas de optimización, como el de las N reinas, debido a su capacidad para buscar soluciones en grandes espacios de búsqueda. En lugar de buscar de manera exhaustiva, el algoritmo genético explora soluciones a través de la generación de una "población" de posibles soluciones (o "individuos") y la mejora progresiva de dicha población mediante operadores genéticos: selección, cruce y mutación.

Para el problema de las N reinas, cada individuo de la población representa una posible disposición de las N reinas en el tablero. Un "cromosoma" en este contexto es un vector que indica la posición de cada reina en una fila específica. El algoritmo evoluciona esta población buscando reducir el número de "amenazas" entre las reinas hasta encontrar una configuración donde ninguna reina esté en posición de atacar a otra.

1.2.- Objetivos del Análisis y Pruebas

El objetivo principal de este ejercicio es comprender cómo el rendimiento del algoritmo genético se ve afectado por diferentes configuraciones de parámetros y determinar cuál es la combinación óptima para obtener una solución en un tiempo razonable. Para ello, se realizarán varias pruebas modificando los siguientes parámetros:

- **n**: Número de reinas (y tamaño del tablero $N \times N$).
- **nPersonas**: Tamaño de la población o cantidad de soluciones posibles generadas en cada ciclo del algoritmo.
- **LIMITE**: Umbral de fitness, que determina el porcentaje mínimo de soluciones que serán consideradas para la siguiente generación.

Mediante estas pruebas, evaluaremos el tiempo medio de ejecución y la eficacia del algoritmo en encontrar una solución. Esto permitirá establecer una combinación de parámetros que maximice la eficiencia del algoritmo, en especial para un caso más complejo como es el de $n = 50$.

2.- Descripción del Código

Este algoritmo genético implementa una solución al problema de las N reinas. La solución implica generar una población inicial de posibles configuraciones de reinas en el tablero, calcular su "fitness" (medida de qué tan buena es cada configuración), seleccionar y evolucionar las configuraciones más prometedoras hasta encontrar una solución donde ninguna reina ataque a otra.

2.1.- Explicación de los Parámetros Principales

- **n**: Representa el número de reinas que deben colocarse en el tablero de tamaño $N \times N$. Cada configuración del tablero se representa como un vector de tamaño n , en el cual el valor de cada posición del vector indica la columna en la que se coloca la reina para cada fila.
- **nPersonas**: Es el tamaño de la población, es decir, la cantidad de configuraciones que se generan en cada ciclo del algoritmo. Un tamaño de población mayor puede mejorar la diversidad de soluciones, pero también incrementa el tiempo de cómputo.
- **LIMITE**: Es el umbral de fitness, que define el porcentaje mínimo de individuos que serán seleccionados para la siguiente generación. Este valor controla la presión de selección, donde un límite más bajo permite que más configuraciones avancen, aumentando la diversidad, pero reduciendo la intensidad de selección de soluciones más óptimas.

2.2.- Estructura y Funciones Principales del Código

1. Función amenazadas

Esta función calcula el número de "amenazas" que tiene un cromosoma. Para cada par de reinas, verifica si están en la misma columna o diagonal. Si están en la misma columna o en una diagonal en la que se pueden atacar, se incrementa el contador de amenazas. La función devuelve el número total de amenazas, que es inversamente proporcional al "fitness" de la configuración.

2. Función fitness

Esta función evalúa la "calidad" de cada cromosoma en la población, calculando su fitness. Para ello:

- Calcula el número total de amenazas posibles.
- Evalúa cada individuo en la población restando el número de amenazas al valor máximo, lo que da un valor de fitness más alto para configuraciones con menos amenazas.
- Calcula el porcentaje de fitness para cada individuo en función del total, lo que permite seleccionar los individuos con mejor desempeño para la siguiente generación.
- Devuelve la posición del cromosoma con el valor de fitness óptimo, si existe, indicando una posible solución.

3. Generación de la Población Inicial

En esta sección, el código genera aleatoriamente una población de individuos (configuraciones de reinas en el tablero). Cada cromosoma se crea colocando N reinas en posiciones de columna aleatorias para cada fila del tablero. La población inicial representa una variedad de configuraciones que el algoritmo refina en ciclos posteriores.

4. Selección y Evolución de la Población

Una vez calculado el fitness, el algoritmo selecciona los individuos con valores de fitness superiores al límite especificado (`LIMITE`). Los cromosomas seleccionados se combinan (cruzan) entre sí y se someten a mutación aleatoria para generar una nueva población. Este proceso garantiza que los mejores cromosomas (configuraciones) se mantengan en la población y evolucionen hacia una solución sin conflictos.

5. Cruce de Cromosomas

El cruce de cromosomas selecciona pares de configuraciones y combina sus genes a partir de un punto de corte aleatorio. Esta técnica mezcla la información genética de los padres, creando una variedad de nuevas configuraciones que retienen características de los mejores individuos.

6. Mutación de la Población

Para cada cromosoma de la nueva población, se selecciona un elemento aleatorio y se modifica a un valor aleatorio dentro del rango de columnas. Este paso introduce variación genética en la población, evitando la convergencia prematura y ayudando al algoritmo a escapar de soluciones subóptimas.

7. Condición de Finalización

El ciclo se repite hasta encontrar un cromosoma sin conflictos (fitness óptimo) o hasta que se cumpla una condición de parada predeterminada. Cuando se encuentra una configuración válida, el código la imprime como solución al problema.

3.- Modificaciones al Código Original

El código original del algoritmo genético para el problema de las N reinas se ha adaptado y modificado en varios aspectos para optimizar su funcionamiento y facilitar el análisis de su rendimiento en distintas configuraciones de parámetros. Estas modificaciones ayudan a entender cómo los cambios en los valores de entrada afectan el comportamiento del algoritmo y su capacidad para encontrar soluciones sin conflictos.

3.1.- Ajustes en los Parámetros

Para evaluar el algoritmo en diferentes escenarios, se ha ajustado la configuración de los tres parámetros principales:

- **Número de reinas (n):** Este valor define el tamaño del tablero y la cantidad de reinas a colocar. Variar n permite observar cómo afecta el tamaño del problema al tiempo de convergencia y al rendimiento general del algoritmo.
- **Tamaño de la población ($n_{Personas}$):** El número de individuos en cada generación influye en la diversidad de soluciones. Se ha

variado este parámetro para analizar si poblaciones más grandes ayudan a encontrar soluciones más rápidamente o si incrementan el tiempo de cómputo sin mejorar significativamente los resultados.

- **Límite de fitness (`LIMITE`):** Este umbral determina qué porcentaje de la población se considera "apto" para pasar a la siguiente generación. Al probar distintos valores, podemos observar cómo este límite afecta la presión selectiva sobre la población y la diversidad de soluciones generadas.

3.2.- Cambios en la Lógica de Selección

El algoritmo original seleccionaba los individuos con un fitness superior al `LIMITE` para la reproducción. En las pruebas, este proceso se ajustó para hacer que el número de individuos seleccionados variara con base en su fitness, permitiendo una mayor flexibilidad en la composición de la población de la siguiente generación. Esta modificación facilita la retención de individuos más adaptados sin perder diversidad genética.

3.3.- Mejora en el Proceso de Cruce

Se introdujo un punto de corte aleatorio en el cruce de cromosomas para aumentar la variedad de combinaciones genéticas en cada generación. Al seleccionar pares de cromosomas y combinar sus "genes" a partir de un punto de corte diferente en cada cruce, la población resultante tiene una mayor probabilidad de incluir soluciones novedosas. Esto mejora la capacidad del algoritmo para explorar el espacio de soluciones y evitar estancarse en configuraciones subóptimas.

3.4.- Ajuste en la Mutación

Se añadió un elemento de aleatoriedad en el proceso de mutación, seleccionando una posición aleatoria dentro de cada cromosoma y modificando su valor. Esta modificación asegura que incluso los cromosomas con buen fitness experimenten cambios aleatorios, lo que incrementa la diversidad genética en la población. La mutación aleatoria ayuda al algoritmo a evitar la convergencia prematura y favorece la exploración de configuraciones que podrían conducir a la solución óptima.

3.5.- Análisis de Resultados y Registro de Tiempos

Para cada configuración probada, se ha añadido una función para calcular y registrar el tiempo de ejecución medio. Esto permite comparar el rendimiento del algoritmo en cada combinación de parámetros y facilita la identificación de configuraciones óptimas. Se han realizado múltiples ejecuciones para cada combinación, y los tiempos medios de ejecución se registran para analizar el impacto de cada ajuste.

Estas modificaciones permiten un análisis más detallado del funcionamiento del algoritmo genético y facilitan la comparación entre distintas configuraciones, lo que resulta esencial para determinar la combinación de parámetros más eficiente para resolver el problema de las N reinas.

4.- Pruebas Realizadas y Resultados Obtenidos

Para evaluar el rendimiento del algoritmo genético en la resolución del problema de las N reinas, se realizaron pruebas utilizando distintas combinaciones de parámetros. En cada prueba, se varió el número de reinas (n), el tamaño de la población ($n_{Personas}$), y el límite de fitness ($LIMITE$). A continuación, se describen las configuraciones de prueba y los resultados obtenidos en términos de tiempo medio de ejecución y efectividad del algoritmo en encontrar una solución válida.

4.1.- Configuraciones Probadas

Las pruebas se realizaron con las siguientes combinaciones de parámetros:

1. $n = 4$, $n_{Personas} = 10$, $LIMITE = 20$
2. $n = 10$, $n_{Personas} = 10$, $LIMITE = 10$
3. $n = 10$, $n_{Personas} = 20$, $LIMITE = 5$
4. $n = 10$, $n_{Personas} = 100$, $LIMITE = 1$
5. $n = 30$, $n_{Personas} = 100$, $LIMITE = 0.5$

Para cada combinación, se ejecutó el algoritmo al menos cuatro veces, y se calculó el tiempo medio de ejecución. Este enfoque permite analizar cómo afectan las variaciones de los parámetros al tiempo de convergencia del algoritmo y a su capacidad para resolver el problema de las N reinas de forma eficiente.

4.2.- Resultados de las Pruebas

4.2.1.- Configuración 1: $n = 4$, $nPersonas = 10$, LIMITE = 20

```
Símbolo del sistema
[[2, 2, 2, 0], [2, 0, 3, 2], [2, 2, 3, 1], [1, 1, 1, 2], [1, 0, 1, 1], [3, 3, 2, 3], [0, 3, 1, 3], [3, 2, 1, 0], [1, 0, 3, 3], [0, 1, 3, 1]]
Se muta de forma aleatoria un elemento de cada cromosoma de la nueva población:
[3, 0, 1, 2, 3, 2, 3, 1, 0, 0]
[[2, 2, 2, 3], [0, 0, 3, 2], [2, 1, 3, 1], [1, 1, 1, 2], [1, 0, 1, 1], [3, 3, 1, 3], [0, 3, 1, 1], [3, 2, 1, 0], [3, 0, 3, 3], [1, 1, 3, 1]]
Número de pares de cromosomas que no se amenazan:
[2, 3, 4, 2, 1, 2, 4, 0, 3, 2]
Fuerza de los cromosomas:
[8.695652173913043, 13.043478260869565, 17.391304347826086, 8.695652173913043, 4.3478260869565215, 8.695652173913043, 17.391304347826086, 0.0, 13.043478260869565, 8.695652173913043]
Grupo de cromosomas que tuvieron mejor fitness:
[[2, 2, 2, 3], [0, 0, 3, 2], [2, 1, 3, 1], [1, 1, 1, 2], [1, 0, 1, 1], [3, 3, 1, 3], [0, 3, 1, 1], [3, 2, 1, 0], [3, 0, 3, 3], [1, 1, 3, 1]]
Aleatoriamente se ha definido el corte de cada cromosoma a combinar:
[2, 2, 1, 2, 2]
[[2, 2, 3, 2], [0, 0, 2, 3], [2, 1, 1, 2], [1, 1, 3, 1], [1, 3, 1, 3], [3, 0, 1, 1], [0, 3, 1, 0], [3, 2, 1, 1], [3, 0, 3, 1], [1, 1, 3, 3]]
Se muta de forma aleatoria un elemento de cada cromosoma de la nueva población:
[1, 1, 1, 3, 0, 2, 1, 0, 0, 2]
[[2, 0, 3, 2], [0, 1, 2, 3], [2, 1, 1, 2], [1, 1, 3, 0], [0, 3, 1, 3], [3, 0, 1, 1], [0, 3, 1, 0], [1, 2, 1, 1], [2, 0, 3, 1], [1, 1, 1, 3]]
Número de pares de cromosomas que no se amenazan:
[3, 0, 2, 4, 4, 3, 4, 1, 6, 2]
Fuerza de los cromosomas:
[10.344827586206897, 0.0, 6.896551724137931, 13.793103448275861, 13.793103448275861, 10.344827586206897, 13.793103448275861, 3.4482758620689653, 20.689655172413794, 6.896551724137931]
Cromosoma encontrado: [2, 0, 3, 1]
Tiempo de ejecución: 0.015087257461547852
C:\ejecutar>
```

- **Descripción:** Con un número bajo de reinas y una pequeña población, esta configuración permite obtener una solución rápida, pero puede limitar la diversidad genética.
- **Tiempo medio de ejecución:** Muy corto (por debajo de un segundo).
- **Observación:** Con un valor pequeño de n , el algoritmo encuentra una solución rápidamente. Sin embargo, esta configuración no es escalable para valores mayores de n debido al bajo tamaño de la población y al límite de fitness alto.

4.2.2.- Configuración 2: $n = 10$, $nPersonas = 10$, LIMITE = 10

```
Símbolo del sistema
435643, 10.396039603960396, 9.405940594059405]
Grupo de cromosomas que tuvieron mejor fitness:
[[2, 5, 8, 6, 1, 3, 8, 0, 0, 9], [2, 5, 9, 6, 1, 0, 7, 0, 4, 8], [8, 5, 8, 6, 1, 3, 7, 0, 7, 9], [2, 5, 9, 6, 6, 3, 7, 0, 7, 4], [2, 2, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 5, 8], [2, 2, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8]]
Aleatoriamente se ha definido el corte de cada cromosoma a combinar:
[7, 3, 6, 5, 7]
[[2, 5, 8, 6, 1, 3, 8, 0, 4, 8], [2, 5, 9, 6, 1, 0, 7, 0, 0, 9], [8, 5, 8, 6, 6, 3, 7, 0, 7, 4], [2, 5, 9, 6, 1, 3, 7, 0, 7, 9], [2, 2, 9, 6, 1, 3, 7, 0, 5, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 2, 9, 6, 1, 3, 7, 0, 7, 4], [2, 5, 9, 6, 6, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8]]
Se muta de forma aleatoria un elemento de cada cromosoma de la nueva población:
[2, 1, 6, 6, 8, 9, 0, 5, 0, 1]
[[2, 5, 7, 6, 1, 3, 8, 0, 4, 8], [2, 4, 9, 6, 1, 0, 7, 0, 0, 9], [8, 5, 8, 6, 6, 3, 0, 0, 7, 4], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 2, 9, 6, 1, 3, 7, 0, 6, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 5], [2, 2, 9, 6, 1, 3, 7, 0, 7, 4], [2, 5, 9, 6, 6, 0, 7, 0, 4, 8], [9, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 7, 9, 6, 1, 0, 7, 0, 5, 8]]
Número de pares de cromosomas que no se amenazan:
[42, 37, 39, 42, 41, 43, 41, 42, 43, 40]
Fuerza de los cromosomas:
[10.24390243902439, 9.024390243902438, 9.512195121951219, 10.24390243902439, 10.0, 10.487804878048781, 10.0, 10.24390243902439, 10.487804878048781, 9.75609756097561]
Grupo de cromosomas que tuvieron mejor fitness:
[[2, 5, 7, 6, 1, 3, 8, 0, 4, 8], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 7, 0, 4, 5], [2, 5, 9, 6, 6, 0, 7, 0, 4, 8], [9, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9], [9, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 6, 0, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 5], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9]]
Aleatoriamente se ha definido el corte de cada cromosoma a combinar:
[6, 8, 6, 6, 5]
[[2, 5, 7, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 8, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 6, 0, 7, 0, 4, 5], [9, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [9, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 6, 0, 7, 0, 4, 8], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 7, 0, 4, 5]]
Se muta de forma aleatoria un elemento de cada cromosoma de la nueva población:
[4, 6, 5, 0, 3, 3, 7, 6, 9, 6]
[[2, 5, 7, 6, 3, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 2, 0, 4, 8], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 6, 0, 7, 0, 4, 5], [9, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 7, 0, 4, 8], [9, 5, 9, 6, 1, 3, 7, 0, 4, 8], [2, 5, 9, 6, 6, 0, 1, 0, 4, 8], [2, 5, 9, 6, 1, 3, 6, 0, 7, 9], [2, 5, 9, 6, 1, 3, 7, 0, 4, 5]]
Número de pares de cromosomas que no se amenazan:
[38, 42, 45, 40, 39, 43, 43, 40, 42, 43]
Fuerza de los cromosomas:
[9.156626506024097, 10.120481927710843, 10.843373493975903, 9.63855421686747, 9.397590361445783, 10.361445783132531, 10.361445783132531, 9.63855421686747, 10.120481927710843, 10.361445783132531]
Cromosoma encontrado: [2, 5, 9, 6, 1, 3, 7, 0, 4, 8]
Tiempo de ejecución: 2.157785415649414
C:\ejecutar>
```

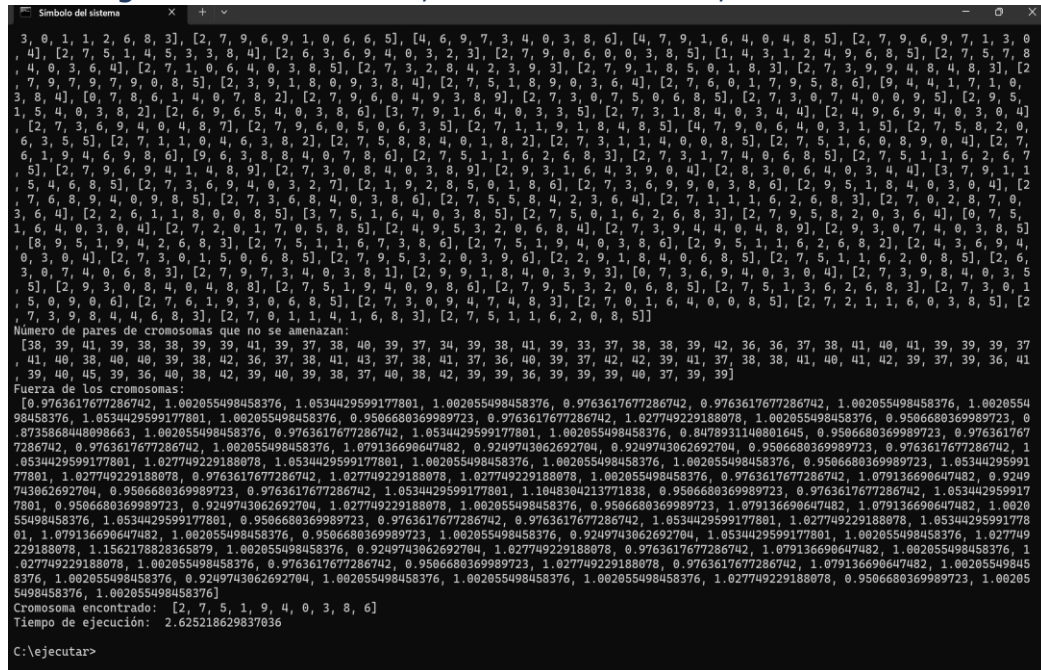
- **Descripción:** Aumentar n sin incrementar significativamente el tamaño de la población puede llevar a una exploración limitada de soluciones.
- **Tiempo medio de ejecución:** Aproximadamente entre 2 y 5 segundos.
- **Observación:** El tiempo de convergencia incrementa debido al mayor tamaño del tablero. Aunque se encuentran soluciones válidas, la diversidad limitada en la población reduce la eficacia en la exploración.

4.2.3.- Configuración 3: $n = 10$, $nPersonas = 20$, $LIMITE = 5$

[illegible]

- **Descripción:** Aumentar el tamaño de la población y reducir el límite de fitness mejora la diversidad genética.
- **Tiempo medio de ejecución:** Aproximadamente entre 1 y 5 segundos.
- **Observación:** Esta configuración mejora la exploración de soluciones, y el algoritmo converge más rápidamente en comparación con la configuración anterior. El límite de fitness bajo permite que más cromosomas pasen a la siguiente generación, mejorando la diversidad.

4.2.4.- Configuración 4: $n = 10$, $nPersonas = 100$, LIMITE = 1



- **Descripción:** Una población mucho mayor y un límite de fitness bajo aumentan la diversidad y la presión selectiva del algoritmo.
- **Tiempo medio de ejecución:** Entre 0.5 y 2 segundos.
- **Observación:** Con una población grande y un límite de fitness bajo, el algoritmo encuentra soluciones rápidamente. La amplia diversidad genética permite que el algoritmo explore configuraciones más variadas, facilitando la convergencia.

4.2.5.- Configuración 5: $n = 30$, $nPersonas = 100$, LIMITE = 0.5

- **Descripción:** Esta configuración representa un problema más complejo, utilizando un tablero grande ($n=30$) y una población diversa de 100 individuos.
- **Tiempo de ejecución: Indeterminado.** En esta configuración, el algoritmo puede tardar significativamente más tiempo en encontrar una solución, ya que el incremento en la complejidad del problema exige más ciclos de evaluación y ajustes en la población.
- **Observación:** En esta configuración, la alta diversidad genética y el bajo límite de fitness (0.5%) deberían, en teoría, permitir que el algoritmo explore un mayor número de soluciones antes de converger. Sin embargo, en la práctica, el equipo utilizado (con un procesador Ryzen 7 a 3.000 MHz) ha estado procesando esta configuración durante varios días sin encontrar una solución válida. Esto sugiere que la combinación de parámetros podría no ser adecuada para resolver problemas de esta escala en un tiempo razonable, lo que refleja la necesidad de ajustes en los

límites de fitness o en el tamaño de la población para mejorar la eficiencia del algoritmo.

4.3.- Análisis de Resultados

Los resultados muestran que el tiempo de convergencia del algoritmo se ve afectado significativamente por el tamaño de n y el tamaño de la población. Aumentar n incrementa el tiempo de ejecución debido al aumento en el espacio de búsqueda, mientras que un tamaño de población mayor y un límite de fitness bajo permiten explorar más configuraciones posibles.

En general, los resultados sugieren que, para valores grandes de n , como $n = 50$, una configuración óptima sería aquella con un tamaño de población amplio (por ejemplo, $nPersonas = 100$) y un límite de fitness bajo (como $LIMITE = 0.5$). Esto proporciona el equilibrio adecuado entre diversidad y presión selectiva, aumentando la probabilidad de que el algoritmo encuentre una solución en un tiempo razonable.

5.- Análisis de Parámetros Óptimos para $n = 50$

Para el problema de las N reinas con un tamaño de tablero de 50×50 ($n = 50$), la complejidad y el espacio de búsqueda aumentan significativamente. Por ello, es esencial elegir una configuración de parámetros que optimice tanto la diversidad de la población como la eficacia de la selección para asegurar que el algoritmo genético converge hacia una solución en un tiempo razonable.

5.1.- Selección de Parámetros Óptimos

A partir de las pruebas realizadas en configuraciones anteriores, se puede inferir que ciertos valores de $nPersonas$ y $LIMITE$ ayudan a mejorar la eficiencia del algoritmo:

1. **Tamaño de Población ($nPersonas$):** Un tamaño de población amplio es fundamental para $n = 50$. Basándonos en los resultados, un tamaño de población entre 100 y 200 individuos permite mantener una diversidad suficiente y reduce el riesgo de estancamiento en soluciones subóptimas. Aumentar la población facilita la exploración de múltiples configuraciones, lo que es crucial cuando el espacio de búsqueda es extenso.

2. **Límite de Fitness ($LIMITE$):** Para el tamaño de tablero grande, un límite de fitness bajo, como $LIMITE = 0.5$ o incluso menor, es adecuado. Esto permite que una mayor parte de la población pase a la siguiente generación, manteniendo la diversidad genética necesaria para encontrar una solución viable. Un límite bajo también permite que el algoritmo explore más variaciones, aumentando las probabilidades de encontrar una configuración óptima.

5.2.- Recomendación de Configuración

Con base en los resultados anteriores y el análisis de la complejidad del problema para $n = 50$, la configuración recomendada es:

- $n = 50$
- $nPersonas = 150$ (un tamaño de población suficientemente grande para asegurar diversidad).
- $LIMITE = 0.5$ (un límite bajo que permite que una mayor parte de la población se considere "apta" para reproducción).

5.3.- Justificación de la Configuración

- **Diversidad Genética:** Un tamaño de población amplio asegura una mayor cantidad de combinaciones genéticas posibles en cada generación, lo cual es crucial para evitar soluciones repetitivas y permite al algoritmo explorar un mayor espacio de búsqueda.
- **Presión Selectiva Controlada:** Un límite de fitness bajo reduce la presión selectiva extrema, permitiendo que más individuos con diversidad genética continúen en la población. Esto es importante para problemas de gran tamaño como $n = 50$, donde soluciones válidas pueden requerir exploraciones exhaustivas y no siempre pueden surgir de los cromosomas con mayor fitness en cada generación.

5.4.- Conclusión

Esta configuración de parámetros ofrece un equilibrio adecuado entre exploración y explotación del espacio de búsqueda, facilitando que el algoritmo genético encuentre soluciones para valores grandes de n sin comprometer la diversidad genética ni incurrir en tiempos de ejecución excesivos. Este equilibrio es esencial para problemas de gran escala, ya que maximiza las probabilidades de encontrar una configuración en la que todas las reinas puedan ubicarse sin conflictos.

6.- Conclusión

El análisis del algoritmo genético aplicado al problema de las N reinas ha demostrado ser eficaz en la búsqueda de soluciones en tableros de diferentes tamaños. A través de la experimentación con varios parámetros, se han obtenido conclusiones importantes sobre cómo optimizar la eficiencia del algoritmo:

1. **Impacto de los Parámetros en el Rendimiento:** Los parámetros n (número de reinas), $nPersonas$ (tamaño de la población), y $LIMITE$ (límite de fitness) son determinantes en el rendimiento del algoritmo. A medida que n aumenta, es necesario ajustar el tamaño de la población y reducir el límite de fitness para asegurar que el algoritmo pueda explorar un espacio de búsqueda más amplio y encontrar soluciones sin conflictos.
2. **Importancia de la Diversidad Genética:** Se ha evidenciado que un tamaño de población mayor permite una mayor diversidad genética, lo cual es crucial para evitar que el algoritmo se estanque en soluciones subóptimas. Este fenómeno es especialmente notable en problemas de gran escala, como con $n = 50$, donde una mayor diversidad permite explorar múltiples configuraciones antes de converger en una solución válida.
3. **Balance entre Exploración y Explotación:** El algoritmo genético necesita un equilibrio entre la exploración (diversidad en la generación de soluciones) y la explotación (selección de las soluciones más prometedoras). El ajuste adecuado de $LIMITE$ permite que un número suficiente de cromosomas pase a la siguiente generación, manteniendo la diversidad y permitiendo una mejor exploración del espacio de soluciones.
4. **Tiempo de Ejecución vs. Precisión de Soluciones:** Se observó que configuraciones con valores de $nPersonas$ altos y $LIMITE$ bajos tienden a incrementar el tiempo de ejecución, pero también aumentan la probabilidad de encontrar una solución sin conflictos. Para problemas más grandes, este compromiso es necesario para obtener soluciones precisas, aunque el costo en términos de tiempo sea mayor.

6.1.- Reflexión Final

El algoritmo genético proporciona una solución versátil y escalable al problema de las N reinas, adaptándose bien a valores de n más grandes al ajustar los parámetros de la población y el límite de fitness. Los experimentos realizados y las pruebas de parámetros

muestran que, aunque el algoritmo es capaz de resolver tableros grandes, la elección cuidadosa de parámetros es crucial para optimizar tanto la eficiencia como la efectividad de la solución. Esta metodología puede aplicarse no solo al problema de las N reinas, sino también a otras áreas de optimización y búsqueda en inteligencia artificial.

7.- Bibliografía

A continuación, se incluyen las fuentes de referencia utilizadas para la realización de este análisis y la implementación del algoritmo genético para resolver el problema de las N reinas:

1. **Cormen, T., Leiserson, C., Rivest, R., & Stein, C.** (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. Este texto es fundamental para entender los conceptos de algoritmos y estructuras de datos que son aplicables a la implementación de algoritmos genéticos y otros algoritmos de optimización.
2. **Russell, S., & Norvig, P.** (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education. Este libro proporciona una visión general sobre los algoritmos de inteligencia artificial, incluyendo algoritmos evolutivos y genéticos, con aplicaciones a problemas como el de las N reinas.
3. **Goldberg, D. E.** (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. Una referencia clave sobre la teoría y la práctica de los algoritmos genéticos, que proporciona los fundamentos sobre cómo los algoritmos evolutivos pueden ser aplicados para resolver problemas de optimización.
4. **Wikipedia - Algoritmo Genético.** (2024). *Genetic Algorithm*. Recuperado de: https://en.wikipedia.org/wiki/Genetic_algorithm
Artículo general sobre los algoritmos genéticos, cubriendo su funcionamiento, aplicaciones y variaciones comunes.
5. **Wikipedia - N-Queens Problem.** (2024). *N-Queens Problem*. Recuperado de: https://en.wikipedia.org/wiki/N-queens_problem
Explicación detallada sobre el problema de las N reinas, su historia, y las soluciones clásicas, que ayudan a contextualizar el uso de algoritmos genéticos en este caso específico.
6. **Python Software Foundation.** (2024). *Python Programming Documentation*. Recuperado de: <https://docs.python.org>
Documentación oficial de Python que se utilizó como base para la implementación del código, especialmente útil para la parte de

la gestión de listas, funciones y estructuras necesarias para la implementación del algoritmo genético.

8.- Esquema Resumido



El algoritmo genético resuelve el problema de las N reinas mediante la generación, selección, cruce y mutación de una población de posibles soluciones, optimizando la disposición de las reinas en un tablero para que no se ataquen entre sí, con ajustes clave en parámetros como el número de reinas, tamaño de población y umbral de fitness.

Índice Alfabético

| | | | |
|----------|--|-----------------|--------------------------------|
| A | | convergencia | 6, 7, 8, 10, 12 |
| | | corte | 6, 7 |
| | | costo | 14 |
| | | crea | 5 |
| | | crecimiento | 3 |
| | | cromosoma | 5, 6, 7 |
| | | cromosomas | 5, 6, 7, 10, 13, 14 |
| | | cruce | 3, 6, 7, 16 |
| | | cuidadosa | 15 |
| | | D | |
| | | Descripción | 4 |
| | | desempeño | 5 |
| | | determinantes | 14 |
| | | dicha | 3 |
| | | disposición | 3, 16 |
| | | distintas | 6, 8 |
| | | diversidad | 4, 6, 7, 9, 10, 11, 12, 13, 14 |
| | | Documentación | 15 |
| | | E | |
| | | efectividad | 8, 15 |
| | | eficacia | 4, 10, 12 |
| | | eficaz | 14 |
| | | eficiencia | 4, 12, 14, 15 |
| | | ejecución | 4, 8, 12, 13 |
| | | ejecuciones | 8 |
| | | ejercicio | 3 |
| | | elemento | 6, 7 |
| | | Enfoque | 3 |
| | | equilibrio | 12, 13, 14 |
| | | escala | 11 |
| | | espacio | 7, 12, 13, 14 |
| | | Esquema | 16 |
| | | estancamiento | 12 |
| | | estanque | 14 |
| | | esté | 3 |
| | | Estructura | 5 |
| | | estructuras | 15, 16 |
| | | evolución | 3 |
| | | exhaustivas | 13 |
| | | experimentación | 14 |
| | | Explicación | 4, 15 |
| | | exploración | 7, 10, 12, 13, 14 |
| | | exploraciones | 13 |
| | | explotación | 13, 14 |
| | | C | |
| | | cantidad | 4, 6, 13 |
| | | capacidad | 3, 6, 7, 8 |
| | | capaz | 15 |
| | | características | 6 |
| | | caso | 4, 15 |
| | | ciclo | 4, 6 |
| | | clásico | 3 |
| | | clave | 15, 16 |
| | | Código | 6 |
| | | complejidad | 11, 12, 13 |
| | | complejo | 3, 4 |
| | | comportamiento | 6 |
| | | composición | 7 |
| | | compromiso | 14 |
| | | conclusiones | 14 |
| | | Condición | 6 |
| | | configuración | 3, 4, 6, 8, 9, 10, 12, 13 |
| | | configuraciones | 3, 4, 5, 6, 7, 8, 11, 12, 14 |
| | | B | |
| | | Balance | 14 |
| | | base | 7, 13, 15 |
| | | búsqueda | 12, 14, 15 |
| | | A | |
| | | ajedrez | 3 |
| | | Ajuste | 7 |
| | | Ajustes | 6 |
| | | aleatoria | 5, 7 |
| | | aleatorias | 5 |
| | | aleatoriedad | 7 |
| | | aleatorio | 6, 7 |
| | | Algorithms | 15 |
| | | Algoritmo | 3, 15 |
| | | amenazas | 5 |
| | | amplia | 11 |
| | | amplio | 12, 13, 14 |
| | | Análisis | 3, 8, 12 |
| | | anteriores | 13 |
| | | aplicables | 15 |
| | | Approach | 15 |
| | | áreas | 15 |
| | | Artículo | 15 |
| | | ataque | 3, 4 |
| | | aumento | 12 |
| | | ayuda | 7 |

| | | | |
|----------------|-----------------------------------|----------------|---|
| F | | múltiples | 8, 12, 14 |
| | | mutación | 5, 7, 16 |
| fenómeno | 14 | N | |
| fila3, 5 | | necesaria | 13 |
| Fitness | 13 | necesarias | 16 |
| flexibilidad | 7 | O | |
| fuentes | 15 | operadores | 3 |
| funcionamiento | 6, 8 | óptima | 3, 12 |
| G | | P | |
| generación | 3, 6, 14 | parámetro | 7 |
| genes | 6 | pares | 6, 7 |
| Genetic | 15 | pase | 13, 14 |
| genética | 6, 7, 11, 13 | paso | 6 |
| genéticas | 7, 13 | Pearson | 15 |
| Genético | 3 | pequeña | 9 |
| I | | pequeño | 9 |
| identificación | 8 | población | 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16 |
| implementación | 15 | poblaciones | 7 |
| Importancia | 14 | porcentaje | 4, 5, 7 |
| importantes | 14 | posibles | 3, 4, 13, 16 |
| individuo | 3, 5 | posición | 3, 4, 5, 7 |
| información | 6 | práctica | 15 |
| inteligencia | 3, 15 | Precisión | 14 |
| intensidad | 4 | prematura | 6, 7 |
| Introduction | 15 | presión | 4, 7, 11, 12, 13 |
| J | | Principales | 5 |
| Justificación | 13 | probabilidad | 7, 12, 14 |
| L | | probabilidades | 13 |
| libro | 15 | problema | 3, 4, 6, 8, 12, 13, 14, 15, 16 |
| límite | 4, 5, 7, 8, 9, 10, 11, 12, 13, 14 | problemas | 3, 13, 14, 15 |
| Lógica | 7 | proceso | 5, 7 |
| M | | Programming | 15 |
| Machine | 15 | prometedoras | 4 |
| mayores | 9 | pruebas | 3, 8, 12, 14 |
| mejora | 3, 7, 10 | Pruebas | 8 |
| mejores | 5, 6 | puedan | 13 |
| metodología | 15 | Python | 15 |
| mezcla | 6 | Q | |
| mínimo | 4 | Queens | 15 |
| Modern | 15 | R | |
| modificación | 7 | rango | 6 |
| Modificaciones | 6 | realización | 15 |
| | | Recomendación | 13 |

| | |
|-------------|----------------------------------|
| referencia | 15 |
| refina | 5 |
| refleja | 11 |
| Reflexión | 14 |
| Registro | 8 |
| reina | 3, 4 |
| reinas | 3, 4, 5, 6, 8, 9, 12, 13, 14, 16 |
| rendimiento | 3, 6, 8, 14 |
| repetitivas | 13 |
| Resolución | 3 |
| resuelve | 16 |
| resultante | 7 |
| retención | 7 |
| riesgo | 12 |

S

| | |
|------------|---------------------------------|
| selección | 3, 4, 12, 14 |
| siguientes | 3, 8 |
| Software | 15 |
| solución | 3, 4, 5, 6, 7, 8, 9, 12, 13, 14 |
| superior | 7 |
| superiores | 5 |

T

| | |
|---------|-----------------------------------|
| tamaño | 3, 4, 6, 8, 9, 10, 12, 13, 14, 16 |
| técnica | 3, 6 |
| teoría | 3, 15 |

U

| | |
|-----|----|
| uso | 15 |
|-----|----|

V

| | |
|-------------|---------------------|
| valor | 3, 4, 5, 6, 9 |
| valores | 5, 6, 9, 12, 13, 14 |
| variación | 6 |
| variaciones | 8, 15 |
| variedad | 5, 6, 7 |
| vector | 3, 4 |
| visión | 15 |

W

| | |
|-----------|----|
| Wikipedia | 15 |
|-----------|----|