Inteligencia Artificial: Trabajo Práctico 3

28 de agosto de 2025

Temas Clave

Estrategias de Búsqueda Local

Métodos para encontrar soluciones óptimas en espacios de estados complejos.

Algoritmos Evolutivos

Técnicas inspiradas en la evolución biológica para la optimización.

Problemas de Satisfacción de Restricciones (CSP)

Enfoques para resolver problemas con un conjunto de restricciones.

Ascensión de Colinas: Mecanismo y Problemas

El algoritmo de Ascensión de Colinas avanza continuamente hacia un "mejor estado" instantáneo, sin mantener un árbol de búsqueda.

Desventajas

- Puede quedar atrapado en máximos locales.
- Dificultad con crestas y mesetas.
- No garantiza encontrar el máximo global.

Áreas Problemáticas

- Máximos locales
- Mesetas (terrazas y mesetas de máximos locales)
- Crestas muy empinadas

Heurísticas en Satisfacción de Restricciones

Heurísticas eficaces y genéricas para problemas de Satisfacción de Restricciones:

1 Chequeo hacia adelante

Reduce el dominio de variables vecinas al asignar un valor.

2 Heurística de grado máximo

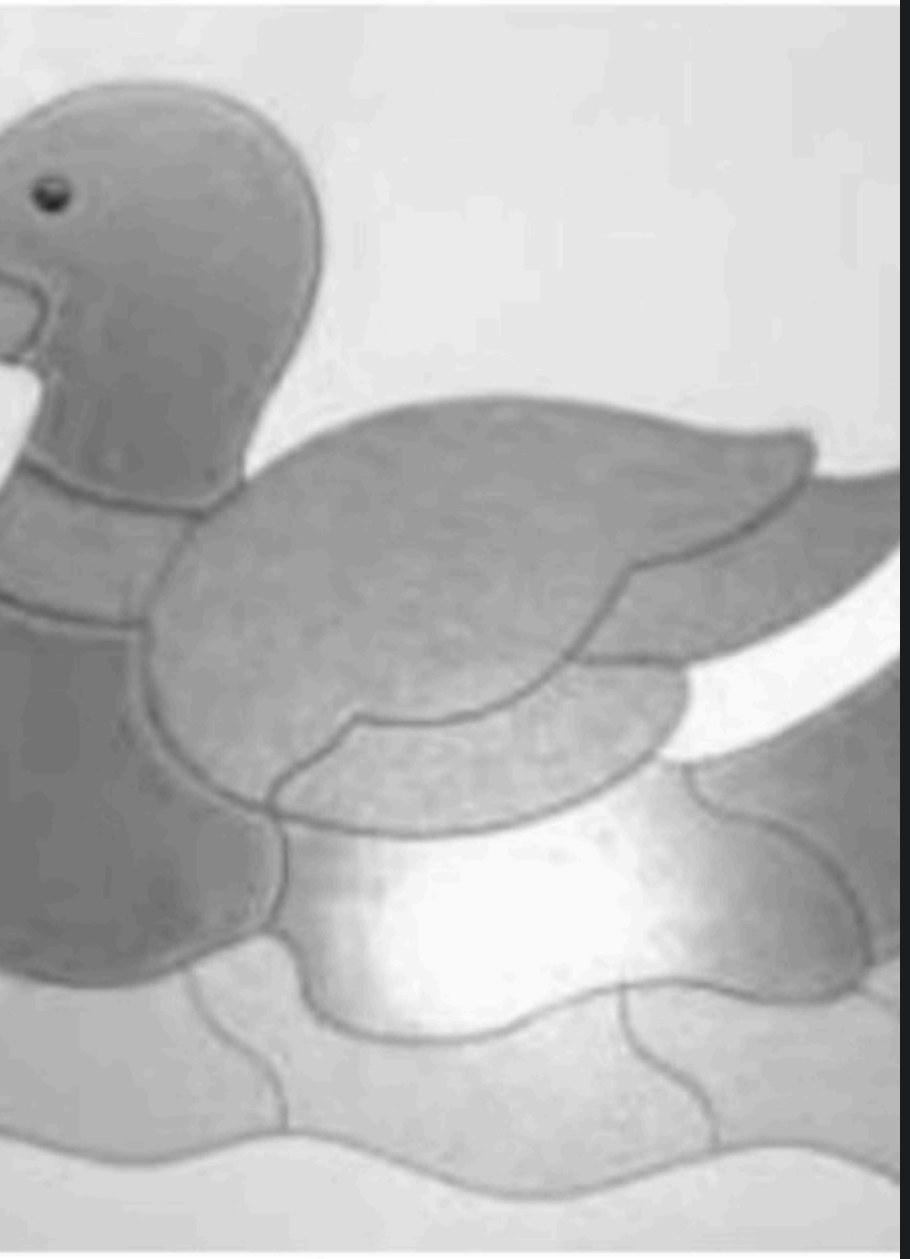
Elige la variable con mayor número de restricciones para evitar conflictos.

3 Heurística de mínimos valores restantes

Selecciona la variable con menos valores legales disponibles.

4 Heurística del valor menos restringida

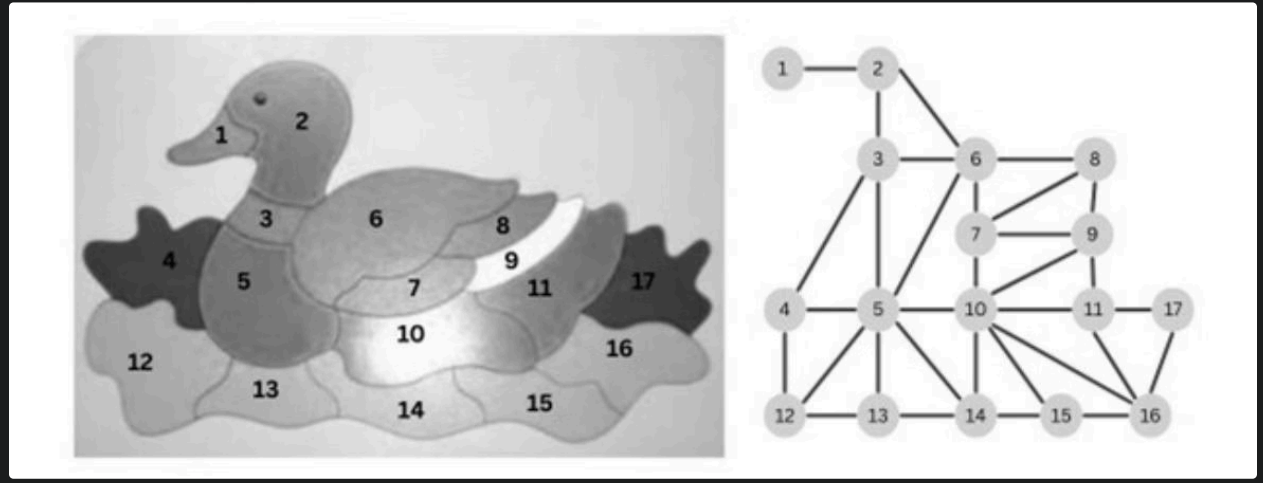
Elige el valor que elimine menos posibilidades para otras variables.



Problema de Coloreado de Rompecabezas

Se busca colorear un rompecabezas con 7 colores distintos, asegurando que ninguna pieza tenga el mismo color que sus vecinas.

Proceso de búsqueda con Comprobación hacia Adelante, empleando una heurística del Valor más Restringido.



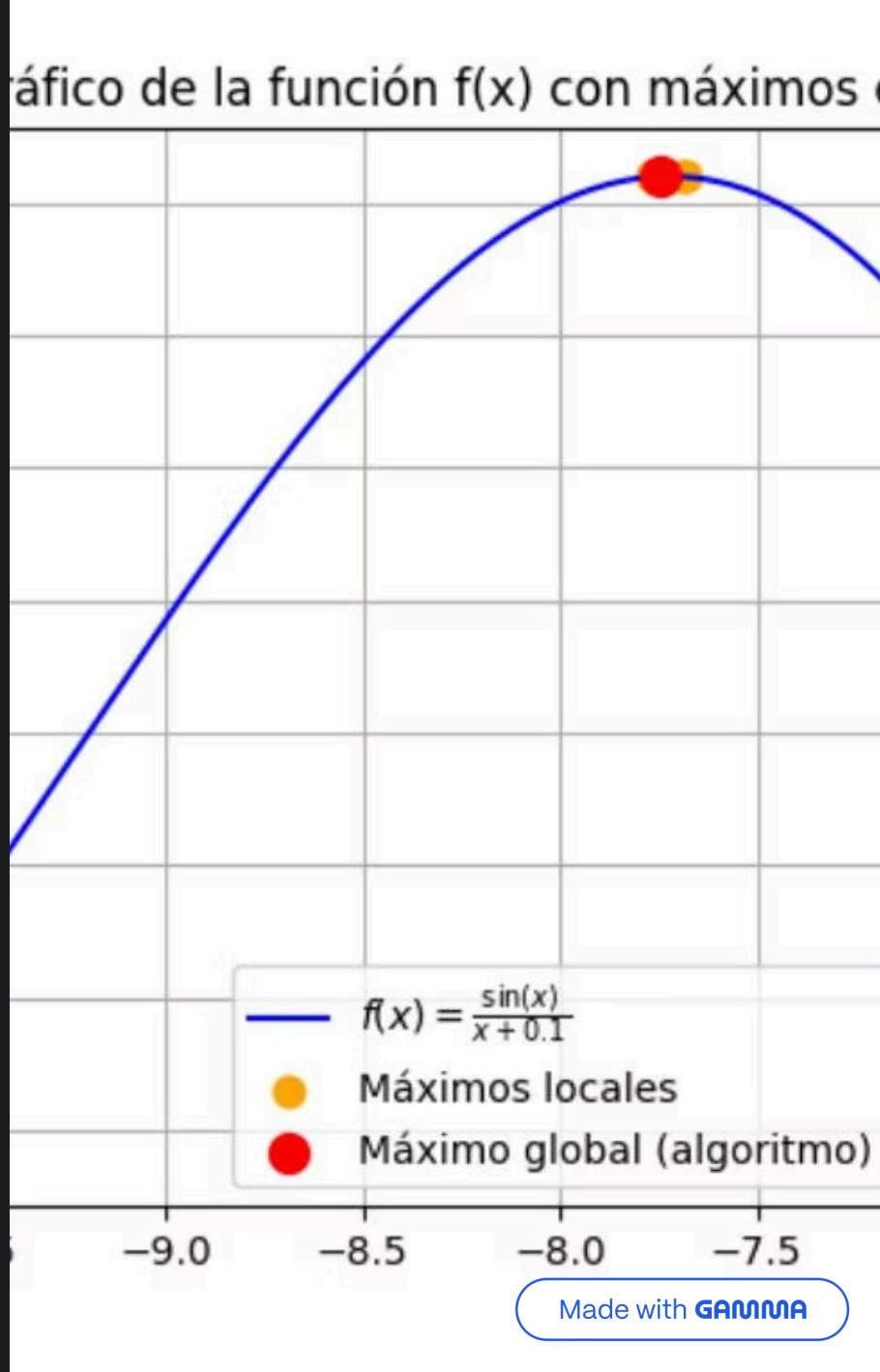
Se observa que solo son necesarios 4 colores para pintar esta imagen.

Implementación: Algoritmo Hill Climbing

Encontrar el máximo de la función $f(x) = \frac{\sin(x)}{x+0.1}$ en $x \in [-10; -6]$ con un error menor a 0.1.

```
def ClimbingHill(f, x0, error, max_iter=100000):
    step = 0.05
    x = x0
    for i in range(max_iter):
        vecinos = [x - step, x + step]
        next_x = max(vecinos, key=f)
        if f(next_x) - f(x) < error:
            return next_x
        x = next_x
```

El máximo global encontrado por el algoritmo es: x = -7.750010920807812 ; f(x) = 0.1300128748659443



Recocido Simulado en Ta-te-ti

Diseño e implementación de un algoritmo de Recocido Simulado (SA) para jugar al Ta-te-ti.

Temperatura Alta

El algoritmo es más permisivo, acepta movimientos "malos" para explorar el espacio de soluciones y escapar de óptimos locales.

Temperatura Baja

La aceptación de jugadas malas casi desaparece; la búsqueda se vuelve codiciosa, solo toma jugadas que mejoran inmediatamente.

La IA puede sorprender con movimientos menos obvios con temperatura alta, mientras que con temperatura baja juega más rígida y predecible.

Componentes del Juego Ta-te-ti con SA

Tablero y Reglas

- `new_board()`: Crea tablero vacío.
- `print_board(b)`: Dibuja el tablero.
- `available_moves(b)`: Posiciones libres.
- `place(b, i, mark)`: Coloca X u O.
- `winner(b)`: Chequea ganador.
- `is_draw(b)`: Verifica empate.
- `copy_board(b)`: Copia el tablero.

Rollouts (Simulaciones)

- `random_policy_move()`: Juega greedy.
- `simulate_from_move()`: Simula partida completa.
- `estimated_value()`: Calcula valor esperado de jugada.

Recocido Simulado (SA)

- `Recocido()`: Elige jugada con SA.
- `T0`: Temperatura inicial.
- `Tf`: Temperatura final.
- `alpha`: Factor de enfriamiento.

Interfaz de Juego

- `ask_move()`: Pregunta jugada al humano.
- `play_human_vs_sa()`: Loop principal del juego.
- `main()`: Punto de entrada.

Algoritmo Genético: Carga de Grúa

Maximizar el precio de las cajas cargadas en una grúa con una capacidad máxima de 1000 kg.

100	300
50	200
115	450
25	145
200	664
30	90
40	150
100	355
100	401
100	395

El algoritmo debe seleccionar las cajas para maximizar el precio sin exceder el límite de peso.

Resultados y Conclusiones

Mejor Solución Encontrada	Peso Total	Precio Total
Cajas a cargar: [1, 5]	964 kg (Límite: 1000 kg)	300 €

Este algoritmo genético demuestra la capacidad de optimizar la carga de la grúa, encontrando una combinación de cajas que maximiza el precio dentro de las restricciones de peso.