Taller: Planificación de Rutas con Búsqueda Anytime (AW-A*)

Clase práctica guiada

Descripción general

En este taller los estudiantes ejecutarán, analizarán y **modificarán** un algoritmo de búsqueda *anytime*, específicamente **Anytime Weighted A* (AW-A*)**, para resolver un caso de planificación de rutas en mapas con obstáculos. El objetivo es que comprendan *cuándo* y *cómo* la búsqueda *anytime* ofrece ventajas frente a enfoques tradicionales.

Objetivos de aprendizaje

Al finalizar, el/la estudiante será capaz de:

- Explicar formalmente las propiedades de un algoritmo *anytime* (interruptibilidad, mejora progresiva y medida de calidad).
- Ejecutar el notebook `anytime_search_lab.ipynb` y **reproducir** resultados base en mapas de distinto tamaño.
- **Instrumentar** experimentos variando tiempo límite, pesos heurísticos y heurística.
- **Modificar** el código para evaluar efectos en tiempo/calidad y extraer conclusiones.
- Analizar comparativamente AW-A*, A* clásico y Greedy Best-First en escenarios con distintas densidades de obstáculos.

Prerequisitos

- Python 3.x con `matplotlib`, `numpy`, `pandas` (u otro entorno que ejecute Jupyter).
- Conocimientos básicos de búsqueda informada (A*, heurísticas) y manejo de notebooks.

Material entregado

- Notebook: `anytime_search_lab.ipynb` (contiene: teoría resumida, implementación de AW-A*, visualización, y bloque de *protocolo experimental* con baselines).

Actividades (paso a paso)

1. Orientación y lectura de código

1.1 En el notebook, localiza:

- `class Node`: ¿qué guarda cada nodo? (`g`, `h`, `f`, puntero a `parent`).
- `heuristic(a,b)`: heurística Manhattan (justifica su admisibilidad en cuadrícula 4-conexa).
- `awastar(...)`: peso w en $f(n) = g(n) + w\,h(n)$, control de `max_time`, poda por cota superior.

1.2 Pregunta guía A:

- ¿Por qué decrementar w tiende a la optimalidad? ¿Qué ocurre si w es muy alto o cercano a 1?

2. Experimentos base

2.1 Ejecuta AW-A* en un mapa 41×41 con `max_time` en {0.2s, 1.0s, 3.0s}.

2.2 Registra la **longitud de la mejor ruta** para cada tiempo. Grafica *calidad vs tiempo* (perfil *anytime*).

2.3 Pregunta guía B: ¿Qué tan buena es la primera solución? ¿Cuánto mejora con tiempo adicional?

3. Modificaciones dirigidas

3.1 Sustituye la heurística Manhattan por **Euclidiana** y repite el experimento:

Listing 1: Heurística Euclidiana (sugerencia mínima)

```
import math
def heuristic(a, b):
    dx, dy = abs(a[0]-b[0]), abs(a[1]-b[1])
    return math.sqrt(dx*dx + dy*dy)
```

3.2 Ajusta pesos: prueba combinaciones de `w_start` $\in \{2.0, 2.5, 3.0\}$, `w_end`$= 1.0$, `w_step`$\in \{0.25, 0.5\}$.

3.3 Aumenta tamaño de mapa a 1500×1300, luego a 2000 X 2000 y compara el % de mejora entre tiempo corto vs largo.

3.4 Pregunta guía C: ¿Qué combinación (heurística, w) entrega mejor *calidad/tiempo*?

4. Estructura del mapa y rutas alternativas

4.1 Usa el generador con diferentes *aperturas extra* (p. ej., 0, 50, 150) para simular *mapa denso* vs *mapa abierto*.

4.2 Compara AW-A*, A* clásico y Greedy Best-First (con el mismo límite de tiempo).

4.3 Pregunta guía D: ¿En qué tipo de mapa AW-A* ofrece mayor beneficio relativo? ¿Por qué?

5. Cierre y reflexión

- **Pregunta guía E**: ¿Cuándo preferirías *anytime* en un sistema real (juego, robótica, logística)?
- **Pregunta guía F**: ¿Qué limitaciones observaste (tiempos, sensibilidad a parámetros, etc.)?

Entregables

E1. Notebook con ejecuciones y **modificaciones** realizadas (celdas claras y comentadas).

E2. Gráficas/tabla de resultados comparativos (perfiles *anytime*, tamaños de mapa, densidad de obstáculos).

E3. Informe breve (1–2 páginas) que responda las Preguntas Guía A–F y proponga al menos **un caso real adicional**.

Rúbrica de evaluación

Criterio	**Excelente (100 %)**	**Bueno (75 %)**	**Aceptable (50 %)**	**Insuficiente (25 %)**
Comprensión teórica	Define y justifica propiedades *anytime* con ejemplos y límites.	Explica propiedades con ejemplos básicos.	Menciona propiedades sin profundidad.	Confusiones o ausencia de explicación.
Implementación y cambios	Ejecuta y modifica código de forma correcta y documentada.	Ejecuta y modifica con detalles menores faltantes.	Ejecuta con pocos cambios.	No ejecuta o provoca errores no resueltos.
Experimentos y análisis	Diseña comparaciones, reporta métricas y discute hallazgos.	Compara con análisis parcial.	Presenta resultados sin análisis.	Resultados incompletos o sin interpretar.
Aplicación al caso	Conecta resultados con el caso (dron) y propone usos realistas.	Conecta parcialmente.	Conexión débil o genérica.	Sin conexión práctica.

Apéndice: Pseudocódigo mínimo de AW-A*

```
# f(n) = g(n) + w * h(n); disminuir w -> 1.0 mientras haya tiempo
best_path <- None; best_len <- +inf
t0 <- now()

for w in [w_start, w_start - step, ..., w_end]:
    if now() - t0 >= max_time: break
    OPEN <- priority queue by f; push(start)
    g_cost[start] <- 0
    CLOSED <- empty set

    while OPEN not empty and now() - t0 < max_time:
        n <- pop_min_f(OPEN)
        if n in CLOSED: continue
        CLOSED.add(n)

        if g(n) + h(n) >= best_len: continue   # poda por cota superior

        if n == goal:
            cand <- reconstruct(n)
            if |cand| < best_len:
                best_len <- |cand|; best_path <- cand
            break    # reducir w y refinar

        for each neighbor v of n (no obst culo):
            ng <- g(n) + cost(n,v)
            if ng < g_cost.get(v, +inf):
                g_cost[v] <- ng
                push(OPEN, v with parent=n, f = ng + w*h(v))

return best_path
```

Nota: En el notebook provisto se incluye instrumentación para registrar la *mejor cota* a lo largo del tiempo y trazar el *perfil anytime*.