Tecnológico de Monterrey

Curso:

Modelación de sistemas multiagentes con gráficas computacionales

Grupo:

302

Estudiante:

Mateo Arminio Castro - A01785572

Juan de Dios Gastélum Flores - A01784523

Juan Pablo Narchi Capote - A01781518

Maestros:

Octavio Navarro Hinojosa

Gilberto Echeverría Furió

Título:

Reto: Movilidad Urbana

Fecha de entrega:

5 de Diciembre 2025

Problema y Propuesta de Solución

Problema: La congestión vehicular urbana y la gestión ineficiente del tráfico son problemas complejos que resultan en pérdidas de tiempo y recursos. Se requiere simular un entorno urbano que incluya intersecciones semaforizadas, obstáculos y comportamientos vehiculares variados (incluyendo conductores erráticos) para analizar el flujo de vehículos desde puntos de origen hasta destinos específicos.

Propuesta de Solución: Se ha implementado una simulación basada en Sistemas Multi-Agentes utilizando el framework Mesa en Python. La solución modela una ciudad como una cuadrícula donde interactúan diversos tipos de agentes:

- **Vehículos:** Agentes inteligentes capaces de navegar, planificar rutas y reaccionar al entorno.
- **Infraestructura:** Semáforos, Carreteras, Obstáculos y Destinos.
- **Agentes Estocásticos:** Se introduce un agente "Borrachito" para probar la robustez del sistema ante comportamientos impredecibles y peligrosos.

Diseño de los Agentes

Para el diseño del agente principal (Car y su variante Borrachito), se utiliza la especificación PEAS:

P - Performance (Métricas de Desempeño): El éxito del agente se evalúa mediante:

1. **Llegada al destino:** Maximizar el número total de vehículos que alcanzan su coordenada meta.
2. **Eficiencia de Flujo:** Minimizar el tiempo que el agente pasa bloqueado. El agente penaliza internamente los estados de inactividad mediante el contador stuck_counter.
3. **Seguridad:** Evitar colisiones con otros vehículos.

E - Environment (Entorno): El agente opera en un entorno de tráfico urbano simulado que contiene:

- **Vías de Tránsito:** Carreteras con direcciones restrictivas (Izquierda, Derecha, Arriba, Abajo).
- **Dispositivos de Control:** Semáforos que cambian dinámicamente de estado.
- **Obstáculos:** Edificios intransitables y bordes del mapa.
- **Otros Agentes:** Vehículos con comportamientos variados que ocupan espacio físico y compiten por las celdas.

A - Actuators (Actuadores): El agente actúa sobre el entorno mediante:

1. **Mecanismo de Desplazamiento:** Método move_to() para cambiar su posición a una celda adyacente.
2. **Dirección:** Capacidad de elegir entre seguir el camino planificado o ejecutar try_lane_change() para moverse lateralmente ante bloqueos.
3. **Salida del Sistema:** Método remove_agent() que retira al agente del entorno al completar su tarea.

S - Sensors (Sensores): El agente percibe el estado del entorno a través de:

1. **Sensor de Proximidad:** Detecta celdas vecinas inmediatas y su contenido.
2. **Visión Frontal:** Escanea hasta 5 celdas adelante en su trayectoria actual para identificar semáforos en rojo o colas de vehículos.
3. **Estimador de Densidad:** Evalúa la cantidad de agentes en un carril específico para tomar decisiones de ruta.
4. **GPS/Navegación:** Acceso a su posición actual y conocimiento de la coordenada de destino.

Arquitectura de Subsunción

El comportamiento del agente Car sigue una arquitectura jerárquica donde las capas de supervivencia y reglas inmediatas inhiben a las capas de planificación.

Capa 0: Recuperación de Fallos

- Condición: if self.crashed:
- Acción: Esperar un tiempo y reaparecer o continuar. Esta capa suprime cualquier otro comportamiento.

Capa 1: Cumplimiento de Objetivo

- Condición: if current_coord == dest_coord:
- Acción: Salir de la simulación. El agente ha cumplido su propósito.

Capa 2: Planificación

- Condición: if self.path is None:
- Acción: Ejecutar algoritmo A* para generar una nueva ruta.

Capa 3: Optimización y Desbloqueo

- Condición: if self.stuck_counter >= 5:
- Acción: Intentar cambiar de carril. Si falla, recalcular ruta con A* considerando los coches como obstáculos.

Capa 4: Seguridad y Reglas de Tráfico

- Condición: Semáforo en rojo o coche enfrente.
- Acción: Detenerse e incrementar stuck_counter.

Capa 5: Navegación Básica (Movement)

- Condición: Camino libre.
- Acción: Moverse a la siguiente celda del camino (self.move_to(next_cell)).

Características del Ambiente

- **Inaccesible:** Aunque el agente tiene acceso técnico al modelo, su diseño limita su percepción a una vecindad local. Métodos clave como look_ahead(steps=5) y calculate_lane_congestion(lookahead=8) demuestran que el agente solo "ve" una porción limitada del entorno y no tiene una visión global omnisciente de todos los vehículos en tiempo real para tomar sus decisiones inmediatas.
- **No determinista:** Las acciones no tienen un resultado único garantizado debido a la aleatoriedad explícita en el código. Ejemplos claros son la probabilidad de fallo/bloqueo en Car.move, el comportamiento aleatorio del agente Borrachito y la probabilidad de choques.
- **No episódico:** La decisión del agente en el paso actual afecta drásticamente sus opciones

futuras. El agente mantiene un estado interno persistente a lo largo del tiempo, como self.path, self.stuck_counter y self.crash_timer, lo que indica que la experiencia no está dividida en episodios aislados sin memoria.

- **Dinámico:** El entorno cambia independientemente de las acciones del agente. Los semáforos cambian de estado basándose en el tiempo global, y otros agentes se mueven simultáneamente, alterando la disponibilidad de las celdas mientras el agente toma decisiones.
- **Discreto:** El ambiente es discreto. El espacio se modela como una cuadrícula con coordenadas enteras finitas. El tiempo avanza en pasos discretos y el conjunto de acciones posibles es finito y fijo.

Conclusiones

La implementación logra simular un sistema de tráfico complejo utilizando reglas locales simples y planificación de rutas. La inclusión del algoritmo A* permite una navegación eficiente, mientras que la lógica de "cambio de carril" añade un nivel de realismo necesario para gestionar la congestión.

La arquitectura de subsunción implementada asegura que la seguridad tenga prioridad sobre el movimiento, pero incluye mecanismos de re-planificación para evitar que los agentes se queden estancados indefinidamente. La adición del agente "Borrachito" valida la capacidad del sistema para manejar excepciones y recuperar el flujo tras incidentes.