UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Master Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universidad Politécnica de Valencia

Simulación de Multitudes mediante Sistemas Multi-Agentes con MESA

TRABAJO SISTEMAS MULTI-AGENTES

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Miquel Gómez

Resumen

Este trabajo presenta el desarrollo e implementación de un sistema de simulación multi-agente para el estudio de dinámicas de evacuación y comportamiento de multitudes. Utilizando el framework MESA en Python, se ha creado un modelo discreto basado en malla que permite analizar el impacto de comportamientos individuales y globales de grupos de personas.

Se han implementado tres tipologías de agentes (educados, agresivos y lentos) que interactúan en diferentes escenarios. Estos escenarios representan espacios y situaciones reales. Los resultados experimentales revelan conclusiones contraintuitivas respecto a la literatura clásica, demostrando que en modelos discretos la agresividad puede mejorar la eficiencia global del sistema de evacuación.

Se presenta el sistema con todos sus componentes, la metodología de evaluación y los resultados obtenidos, destacando las implicaciones prácticas para el diseño de espacios seguros y eficientes.

Palabras clave: Sistemas Multi-Agentes; Agentes; Simulación; Multitudes; Simulación de multitudes.

Índice general

CAPÍTULO 1
Introducción

La gestión de multitudes y la seguridad en grandes eventos son preocupaciones crecientes en la planificación urbana y arquitectónica. Entender cómo se mueven las personas, cómo reaccionan ante emergencias y cómo interactúan entre sí es fundamental para diseñar espacios seguros y eficientes.

La simulación informática surge como una alternativa ética, económica y segura a los experimentos con personas reales, permitiendo probar escenarios hipotéticos de riesgo sin poner en peligro vidas humanas.

Este trabajo surge de la asignatura de Sistemas Multi-Agente, con el objetivo de experimentar con el framework MESA en Python para crear un entorno de experimentación sobre multitudes. Se centra en el desarrollo y análisis del sistema, así como en la evaluación de su comportamiento bajo diferentes configuraciones y escenarios.

1.1 Alcance y Objetivos

El objetivo principal de este trabajo es desarrollar un entorno de simulación multiagente (MAS) flexible utilizando la librería MESA en Python, que permita estudiar el impacto de los comportamientos individuales heterogéneos en la dinámica macroscópica de una multitud.

- **Abstracción frente a Precisión Física**: Este proyecto **no pretende** crear una herramienta de ingeniería civil certificable bajo la norma ISO 20414 para validación legal de planos, ni replicar con exactitud milimétrica unidades físicas reales.
- **Enfoque en Tendencias y Comportamiento**: El objetivo es simular tendencias emergentes y relaciones causales (p.ej. "¿Cómo afecta un 10 % de agentes agresivos al tiempo total de evacuación?") en lugar de predecir tiempos exactos de evacuación.
- **Tipología de Agentes**: Se busca simular la variabilidad de las personas usando perfiles de agentes (*Polite*, *Aggressive*, *Slow*) para observar la composición de la multitud altera el comportamiento de esta.

1.1.1. Tipos de situaciones a modelar

El sistema se centra en el estudio de situaciones de evacuación de emergencia, donde la rapidez y eficiencia del movimiento colectivo son críticas. Se pretende analizar cómo diferentes configuraciones de agentes y escenarios afectan métricas clave como el tiempo total de evacuación, la tasa de flujo y la congestión.

Estos sistemas son relevantes en contextos como:

- **Eventos Masivos**: Conciertos, festivales y eventos deportivos donde grandes multitudes deben ser gestionadas de manera segura.
- **Infraestructuras Públicas**: Aeropuertos, estaciones de tren y centros comerciales que requieren planes de evacuación efectivos.
- **Diseño Urbano**: Planificación de espacios públicos para optimizar el flujo peatonal y minimizar riesgos en situaciones de emergencia.

Ahora, el sistema también está dotado de la capacidad de simular **flujos continuos de personas**, pero este aspecto no se ha explorado en profundidad en este trabajo por limitaciones de alcance.

El enfoque principal sigue siendo la evacuación de emergencia, sin embargo, el sistema es lo suficientemente flexible como para permitir futuras investigaciones en escenarios de flujo diario y confort peatonal. La idea sigue siendo validar si un espacio es seguro y eficiente bajo diferentes condiciones de multitud. Permitiendo modelar de forma sencilla distintos escenarios y composiciones de agentes.

1.2 Uso de MESA

MESA es un framework de código abierto en Python diseñado para facilitar la creación, simulación y visualización de modelos basados en agentes. Proporciona una estructura modular que permite a los desarrolladores definir agentes, entornos y reglas de interacción de manera sencilla.

Además, una vez el sistema está implementado, MESA ofrece herramientas integradas para la recolección de datos y la visualización en tiempo real, lo que facilita el análisis de los resultados de las simulaciones.

En este trabajo, MESA se utiliza como la base para construir el modelo de simulación de multitudes, aprovechando sus capacidades para gestionar agentes heterogéneos y sus interacciones en un entorno discreto basado en malla.

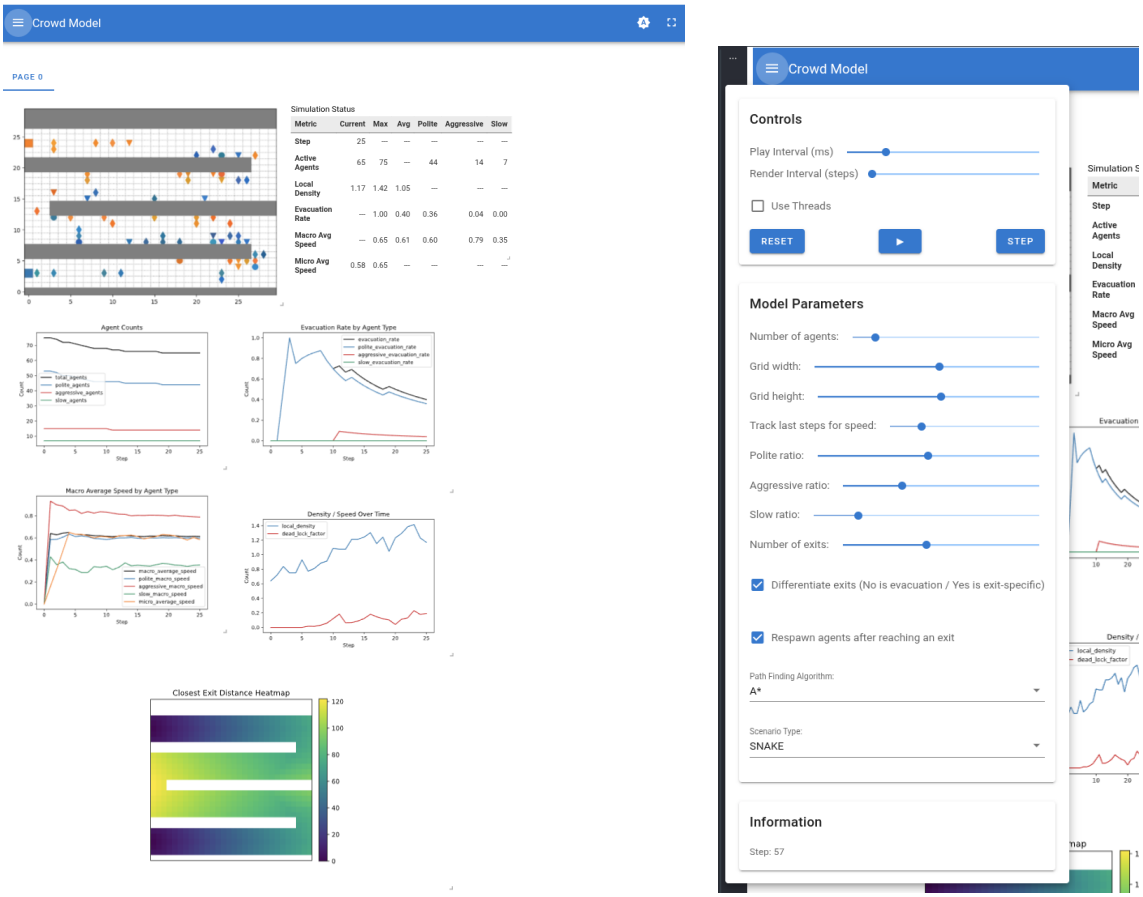
Se ha implementado todo el sistema siguiendo la documentación oficial de MESA [**mesa**] y adaptando sus componentes a las necesidades específicas del estudio de multitudes.

1.3 Experimentación

El trabajo incluye un notebook de Jupyter que permite ejecutar simulaciones con diferentes configuraciones y escenarios, cambiarlos y visualizar los resultados y las estadísticas en tiempo real.

Este notebook facilita la experimentación al permitir modificar parámetros como la densidad de agentes, la proporción de tipos de agentes y las características del entorno. Además, incluye gráficos y tablas que resumen las métricas clave de cada simulación, permitiendo un análisis comparativo entre diferentes configuraciones.

Cualquier persona interesada puede clonar el repositorio del proyecto, instalar las dependencias necesarias y ejecutar el notebook para replicar los experimentos o realizar nuevas pruebas con diferentes parámetros.



(a) Simulación en ejecución. **(b)** Interfaz de usuario del notebook.

Figura 1.1: Capturas de pantalla del sistema de simulación final.

1.4 Estructura del Documento

El documento se estructura en: estado del arte, donde se revisan las tecnologías actuales; análisis del problema, donde se define la estrategia de modelado; evaluación experimental, detallando métricas y escenarios; implementación técnica; y finalmente los resultados y conclusiones obtenidos.

CAPÍTULO 2
Estado del Arte

La simulación de multitudes ha evolucionado de ser una disciplina académica minoritaria, a convertirse en una herramienta crítica para garantizar la seguridad, eficiencia y confort en infraestructuras modernas.

Este capítulo evalúa el estado del arte actual, dividiendo el análisis en los objetivos generales que siguen los desarrolladores de estos sistemas: los marcos regulatorios vigentes y las arquitecturas de micro-comportamiento utilizadas para modelar la navegación de agentes.

2.1 Objetivos Globales de la Simulación

En el panorama actual de muchos sectores, la simulación de multitudes responde a regulaciones y consideraciones económicas, clasificándose principalmente en Diseño Basado en Prestaciones (Performance-Based Design) y Análisis de Nivel de Servicio.

2.1.1. Diseño Basado en Prestaciones (PBD) y Seguridad

Históricamente, los códigos de edificación eran toscos, dictando reglas rígidas sobre anchos de escalera y distancias de evacuación. El enfoque PBD permite diseños flexibles siempre que se demuestre, mediante simulación, que se cumplen los criterios de seguridad [1].

La métrica fundamental en PBD es la relación entre el Tiempo Disponible para la Evacuación Segura (ASET) y el Tiempo Requerido para la Evacuación Segura (RSET). Para que un diseño sea válido, debe cumplirse que:

$$ASET > RSET + Margen\_de\_Seguridad \tag{2.1}$$

Mientras que el ASET depende de la dinámica del fuego (toxicidad, calor, humo), el RSET es calculado por el simulador de multitudes y se compone de:

- **Tiempo de Detección** y **Tiempo de Notificación**: Darse cuenta del peligro y alertar.
- **Tiempo de Pre-evacuación**: El intervalo entre la alarma y el primer movimiento del agente.
- **Tiempo de Viaje**: La fase física del movimiento desde el origen hasta una zona segura. Este es el componente principal que modela el sistema desarrollado en este trabajo.

2.1.2. Nivel de Servicio (LOS) y Confort

Para escenarios no urgentes sin peligros (más "Daily Life"), el objetivo principal es evaluar el confort peatonal. El estándar de referencia es la escala de Nivel de Servicio (LOS) propuesta por Fruin [2], que clasifica el flujo peatonal en seis niveles (A a F) basándose en la densidad local (personas/m^2):

- **LOS A (Flujo Libre)**: Densidad baja ($< 0{,}31 p/m^2$), los peatones eligen libremente su velocidad.
- **LOS F (Congestión)**: Densidad crítica ($> 2{,}17 p/m^2$), flujo interrumpido y contacto físico inevitable.

2.2 Marcos Regulatorios y Validación

Para que un modelo de simulación sea considerado una herramienta válida y no un mero ejercicio teórico, debe adherirse a estándares de validación internacionales.

- **ISO 20414:2020**: Esta norma establece protocolos rigurosos de verificación y validación para modelos de evacuación. Exige pruebas de componentes (p.ej., verificar que un agente se mueve a la velocidad asignada), verificación funcional (capacidad de flujo en puertas, contraflujo en pasillos) y validación cualitativa de comportamientos emergentes como la formación de arcos en las salidas [3].
- **NIST y SFPE**: El National Institute of Standards and Technology y la Society of Fire Protection Engineers proporcionan los datos demográficos y de comportamiento estándar (velocidades según edad, dimensiones corporales) que deben utilizarse para configurar los agentes [4].

Dado el alcance académico de este trabajo, no se pretende certificar el modelo bajo ISO 20414, pero se han seguido sus directrices para asegurar una base sólida y reproducible.

2.3 Arquitecturas de Navegación y Micro-comportamientos

La lógica de un simulador de multitudes reside en sus algoritmos de navegación local y resolución de conflictos. Existen dos tipos principales en la literatura:

2.3.1. Modelos de Espacio Continuo

Dominantes en la industria de la animación y la robótica por su fidelidad visual.

- **Social Force Model (SFM)**: Propuesto por Helbing [5], trata a los agentes como partículas sometidas a fuerzas newtonianas. Combina una fuerza impulsora hacia la meta con fuerzas repulsivas socio-psicológicas para mantener la distancia personal.
- **Optimal Reciprocal Collision Avoidance (ORCA)**: Opera en el espacio de velocidades para garantizar matemáticamente trayectorias libres de colisiones [6]. Aunque eficiente, puede resultar en comportamientos demasiado 'perfectos' o robóticos.

2.3.2. Modelos Discretos y Grid-Based

Los modelos basados en Autómatas Celulares (CA) o mallas discretas dividen el espacio en celdas. Son la base del sistema implementado en este trabajo. El desafío principal es la resolución de conflictos (cuando dos agentes compiten por la misma celda). El estado del arte actual emplea *Floor Fields* (campos de potencial) estáticos y dinámicos para guiar a los agentes, y utiliza teoría de juegos o heurísticas de "paciencia"para resolver bloqueos, superando las reglas simples de exclusión [7].

CAPÍTULO 3
Análisis del problema

El desarrollo de simuladores de multitudes implica equilibrar el realismo físico con la complejidad computacional. Dado el objetivo de estudiar comportamientos emergentes en grandes grupos, se requiere una aproximación que permita simular cientos de entidades interactuando en tiempo real o cercano al tiempo real, por lo que se prioriza la ejucución del trabajo para la asignatura antes que la precisión milimétrica.

Además, el framework MESA en Python, aunque flexible y modular, no está optimizado para simulaciones de alta fidelidad física. Por lo tanto, esto nos deja como mejor opción un modelo discreto basado en rejilla que simplifica la representación espacial y las interacciones entre agentes.

3.1 Modelo: Discrete Grid-Based

Para llevar a cabo las simulaciones, se ha optado por un **Modelo Discreto basado en Rejilla (Grid-Based)**, implementado sobre el framework **MESA** en Python.

Comparado con los modelos planteados en el estado del arte, esta elección presenta las siguientes cualidades:

1. **Espacio Discreto**: El entorno se divide en celdas cuadradas, donde cada celda haremos que solo pueda contener a un agente, a una pared (obstáculo) o una salida. Esto simplifica la detección de colisiones y elimina los costosos cálculos de geometría compleja.

2. **Tiempo Discreto**: La simulación avanza en pasos discretos o "steps". La velocidad se modela probabilísticamente: un agente con "mayor velocidad"simplemente tiene una mayor probabilidad de moverse en un step dado.

3. **Navegación**: La estructura de malla facilita la implementación directa de algoritmos de búsqueda de caminos en grafos como BFS y A*, esenciales para la planificación de rutas de los agentes. Con estos algoritmos, se pueden modelar las rutas óptimas hacia las salidas teniendo en cuenta los obstáculos.

3.2 Agentes: Comportamiento

Si nos fijamos en análisis reales de multitudes, estas son de todo menos homogéneas. Para obtener resultados significativos, el sistema debe ser capaz de modelar distintos perfiles de agentes que interactúen entre sí.

En concreto, para este trabajo se han definido tres tipos de agentes con comportamientos diferenciados que pretenden agrupar las características más relevantes observadas en la gente:

- **Polite (Educados)**: Serían la gente por defecto, educados y cooperativos. Deben ser capaces de ceder el paso y esperar, evitando conflictos activos y no imponerse ante los demás.
- **Aggressive (Agresivos)**: Modelan la competencia y el pánico. Sería la gente que corre y se pone por delante. Su lógica ha de priorizar su movimiento sobre el de los demás, ocupando espacios libres sin consideración.
- **Slow (Lentos)**: Modelan la diversidad física, gente con dificultades, niños o personas mayores. Actúan como obstáculos dinámicos y permiten estudiar el impacto de los usuarios más vulnerables en el flujo general.

Estos agentes serán muy simples en su lógica, interacción entre ellos y en la composición de la multitudes pero con pequeñas variaciones que los cualifican. Esto permitirá observar dinámicas emergentes interesantes. En concreto, vamos a evaluar comportamientos es choques de flujo, cuellos de botella y congestión en pasillos estrechos.

3.3 Movimiento

Para el movimiento de los agentes, se implementa una lógica simple basada en la búsqueda del camino óptimo hacia la salida más cercana utilizando algoritmos como A* o BFS.

De forma eficiente y únicamente al inicio de la simulación, el sistema creará 'rutas precomputadas' en la que los agentes sepan donde deben ir en cada momento, teniendo en cuenta los obstáculos y las paredes.

Se menciona esto porque es un aspecto crucial del sistema, ya que los agentes agentes no actualizarán sus rutas de forma dinámica en función de la posición de los demás agentes. Se seguirán rutas precomputada hacia la salida más cercana.

Se pretende que los agentes resuelvan estas situaciones con una serie de reglas y comportamientos adicionales.

3.3.1. Dead locks

En las simulaciones basadas en mallas discretas, un desafío común es la aparición de **dead locks** o bloqueos mutuos entre agentes. Estos ocurren cuando varios agentes intentan moverse en direcciones opuestas en un espacio limitado, resultando en una situación donde ninguno puede avanzar: Uno va a la celda del otro y viceversa, pero para podes moverse, la celda debe estar libre.

Para mitigar estas situaciones, se definen una serie de reglas y mecanismos dentro de la lógica de los agentes. Se mantendrá un contador de bloqueo que se incrementa cada vez que un agente no puede moverse y el comportamiento del agente cambiará ligeramente en función de este contador.

Cada tipo de agente se verá más o menos afectado por los dead locks, y lo que provocará que los agentes dejen de ir en la dirección más próxima a la salidas y se muevan de forma aleatoria a celdas vecinas para intentar aliviar la congestión y / o encontrar una ruta alternativa.

3.3.2. Preferencias

De forma general se hará que los agentes escaneen el entorno en una dirección concreta, de forma que en caso de empate, se priorice la celda que esté en esa dirección. Esto simula el comportamiento humano de 'siempre ir a la derecha' o 'seguir recto'.

Además, los agentes también pueden tener preferencias de salida. En lugar de dirigirse siempre a la salida más cercana, algunos agentes pueden tener una salida asignada o preferida.

Esto simula situaciones donde las personas tienen conocimiento previo del entorno o están siguiendo 'por donde han entrado'. Esto puede afectar la dinámica de la multitud, especialmente en escenarios con múltiples salidas alejadas entre ellas.

3.3.3. Altas densidades

Cuando la densidad de agentes aumenta, el movimiento se vuelve más difícil y la gente no camina de forma fluida.

Para simular este fenómeno, se implementa una mecánica donde los agentes tienen una probabilidad creciente de no moverse en función del número de vecinos que tengan en las 8 celdas adyacentes. Esto simula el efecto de 'pisar con cuidado' o 'moverse lentamente' en situaciones de alta densidad.

En concreto, esta probabilidad solo empezará a contar cuanto hayan más de 3 vecinos, y no afectará igual a todos los tipos de agentes. Los agresivos serán menos propensos a verse afectados, mientras que los amables serán más sensibles a la congestión.

CAPÍTULO 4
Evaluación

La validación y análisis del modelo propuesto se realiza mediante una serie de experimentos controlados. En este capítulo se detallan las métricas diseñadas para cuantificar el comportamiento de la multitud y la metodología de los experimentos.

4.1 Métricas de Evaluación

Se han implementado monitores de datos (Data Collectors, clase de MESA) que registran paso a paso el estado de la simulación.

Aunque las unidades son abstractas (celdas, ticks), estas métricas son análogas a las utilizadas en estudios de seguridad profesional:

- **Tiempo de Evacuación (Total Steps)**: Cantidad de ticks necesarios para que el último agente abandone el escenario. Representa el RSET (Required Safe Egress Time) y es la métrica principal de eficiencia.
- **Tasa de Evacuación (Flow Rate)**: Cantidad de agentes que alcanzan una salida por tick. Permite visualizar la constancia del flujo e identificar cuellos de botella (caídas en el flow rate).
- **Densidad Local y Crowding**: Se calcula para cada agente la ocupación de sus 8 celdas vecinas. El promedio global de esta métrica indica el nivel de congestión del sistema.
- **Factor de Bloqueo (Deadlock Factor)**: Métrica específica diseñada para este sistema discreto. Mide la proporción de agentes que *intentaron* moverse pero no han podido debido a que sus celdas objetivo estaban ocupadas. Un alto factor de bloqueo indica fricción ineficiente o colapso del flujo.
- **Velocidad Macro y Micro**:
  - *Velocidad Macro*: Velocidad promedio de todo el conjunto de agentes en el sistema a lo largo de toda la simulación.
  - *Velocidad Micro*: Velocidad individual efectiva 'actual'. Mide la velocidad media pero solo teniendo en cuenta los últimos ticks.

Luego, también se recogen métricas de cada una por cada tipo de agente (p.ej. tiempo de evacuación medio de los agresivos vs educados) además de las globales.

4.2 Diseño de Experimentos

Se han diseñado escenarios específicos para poner a prueba distintas hipótesis sobre la dinámica de multitudes. Aunque el sistema que se ha creado soporta simulación de flujo continuo, este estudio se centra exclusivamente en el **Escenario de Evacuación**, donde el objetivo es vaciar el recinto.

4.2.1. Escenarios

En total, se han definido 6 escenarios diferentes, de los cuales nos interesan 4 para este trabajo:

- **OPEN:** Un espacio abierto sin obstáculos. Hay salidas en los bordes de la malla. No es de mucho interés para este trabajo, pero sirve como referencia y espacio de experimentación.
- **MALL:** Pretende simular un centro comercial donde se juntan varios pasillos con tiendas al rededor. Los agentes pueden moverse por los pasillos y por el anillo exterior, pero no pueden atravesar las tiendas. Las salidas están de dos a dos en los bordes de la malla.
- **CORRIDOR:** Similar a MALL, pretende simular la intersección de varios pasillos con poco espacio para moverse. Las salidas están de dos a dos en los finales de cada pasillo.
- **SEATS:** Pretende simular un auditorio, cine o concierto, en el que todas las salidas están en un lado y el resto de la sala está llena de filas de asientos con 'pasillos' entre medias.
- **SNAKE:** Simula un pasillo zigzagueante con una o dos salidas en los extremos. La idea detrás de este escenario es ver el comportamiento de los agentes es espacios estrechos y con giros. Podrían simular una cola de personas.
- **RANDOM:** Por último, este escenario genera obstáculos (paredes, círculos y cuadrados) de forma aleatoria en la malla. Las salidas están en los bordes de la malla. Al igual que OPEN, este escenario no es de mucho interés para el trabajo, pero sirve como referencia y espacio de experimentación.

La cantidad de salidas en cada escenario se puede configurar, yendo desde una sola salida hasta una cierta cantidad definida para cada uno. El tamaño de estos se puede cambiar y los obstáculos y salidas se adaptan de forma dinámica al nuevo tamaño.

4.2.2. Variables Experimentales

En cada escenario, se modifican las siguientes variables independientes para observar su impacto en las métricas:

- **Densidad Poblacional**: Se varía el número de agentes iniciales (de 10 a 300) para observar la transición de flujo libre a congestión.
- **Composición (Polite vs Aggressive vs Slow)**: Se alteran los ratios de agentes para responder preguntas como: ¿Es una multitud cooperativa siempre más rápida? ¿Cómo penalizan los agentes lentos al grupo?

- **Estrategia de Navegación (A* vs BFS)**: Se comparan algoritmos para ver si el cálculo de rutas óptimas (A*) compensa su coste computacional frente a la exploración simple (BFS).
- **Configuración de Salidas**: Se prueba el impacto de asignar salidas fijas (preferencias) frente a dejar que los agentes elijan la más cercana (oportunismo).

4.3 Deadlocks

Los deadlocks serán estudiados mediante la métrica de **Factor de Bloqueo**, observando su evolución temporal y su correlación con la densidad y composición de la multitud.

Se analizarán situaciones donde los deadlocks son más prevalentes, como en pasillos estrechos o intersecciones, y se evaluará la efectividad de las estrategias implementadas para mitigarlos.

4.4 Flujos

Como se ha mencionado, el sistema soporta la simulación de flujos continuos de personas, lo que permite ver como de transitable es un espacio y como le afectan variables como la densidad o la configuración del espacio.

Ahora, este aspecto no se ha explorado en profundidad en este trabajo por limitaciones de alcance. Sin embargo, el sistema es lo suficientemente flexible como para permitir futuras investigaciones en escenarios de flujo diario y confort peatonal.

Por ejemplo, se pueden re hacer los experimentos mencionados anteriormente pero con la opción de *reaparición de agentes* activada. Luego, tras una cantidad decente te steps, el flujo se estabiliza y se pueden recoger métricas como la densidad media, velocidad media y ritmo de evacuación.

Durante el desarrollo, también se ha visto que estos sistemas tienden a oscilar, creando tapones de agentes que luego se disipan, y que pueden ser estudiados con las métricas implementadas.

CAPÍTULO 5
Implementación

El sistema ha sido implementado utilizando Python y la librería **MESA**. Como se ha mencionado, MESA es un framework modular para el modelado basado en agentes y es bastante popular porque permite una rápida prototipación, una fácil extensibilidad de comportamientos y la recolección eficiente de métricas estadísticas.

Los principales componentes que hay que definir en MESA son los mencionados hasta ahora: el modelo (entorno) y los agentes (comportamiento). Además, MESA ofrece herramientas para la visualización y recolección de datos que se han aprovechado en este trabajo.

5.1 Visualización y Recolección de Datos

MESA incluye un módulo de visualización basado en navegador web que permite observar la simulación en tiempo real.

Aprovechamos esta funcionalidad para crear una interfaz interactiva donde se pueden ajustar parámetros como la cantidad de agentes, la proporción de tipos de agentes o las características del entorno antes de iniciar la simulación entre otros.

Además, para hacer visibles los agentes, coloreamos de diferentes tonos de ciertos colores y formas según su tipo: los agentes *Polite* serán rombos de color azul, los *Aggressive* serán triángulos rojos y los *Slow* círculos verdes. También, los obstáculos y las salidas serán cuadrados, los obstáculos se representarán en gris oscuro y las salidas cada una de un color. En las simulaciones en las que los agenges tengan una salida favorita, se usarán los colores de cada una en lugar de los colores por defecto.

Haremos gráficas en tiempo real de las métricas más importantes, una tabla resumen de los datos actuales de la simulación y una visualización del entorno con la distancia a la salida más cercana de cada celda.

En la Figura 1.1 se pueden ver capturas de pantalla del sistema de simulación final con ejemplos de lo mencionado.

5.2 Estructura del Modelo

En MESA, un modelo es una clase que hereda de `mesa.Model` y que contiene la lógica principal de la simulación.

En concreto, los modelos de MESA se encargaran principalmente de inicializar el entorno, crear los agentes, definir la función `Step` que mueve a los agentes y actualizar y controlar el estado del entorno.

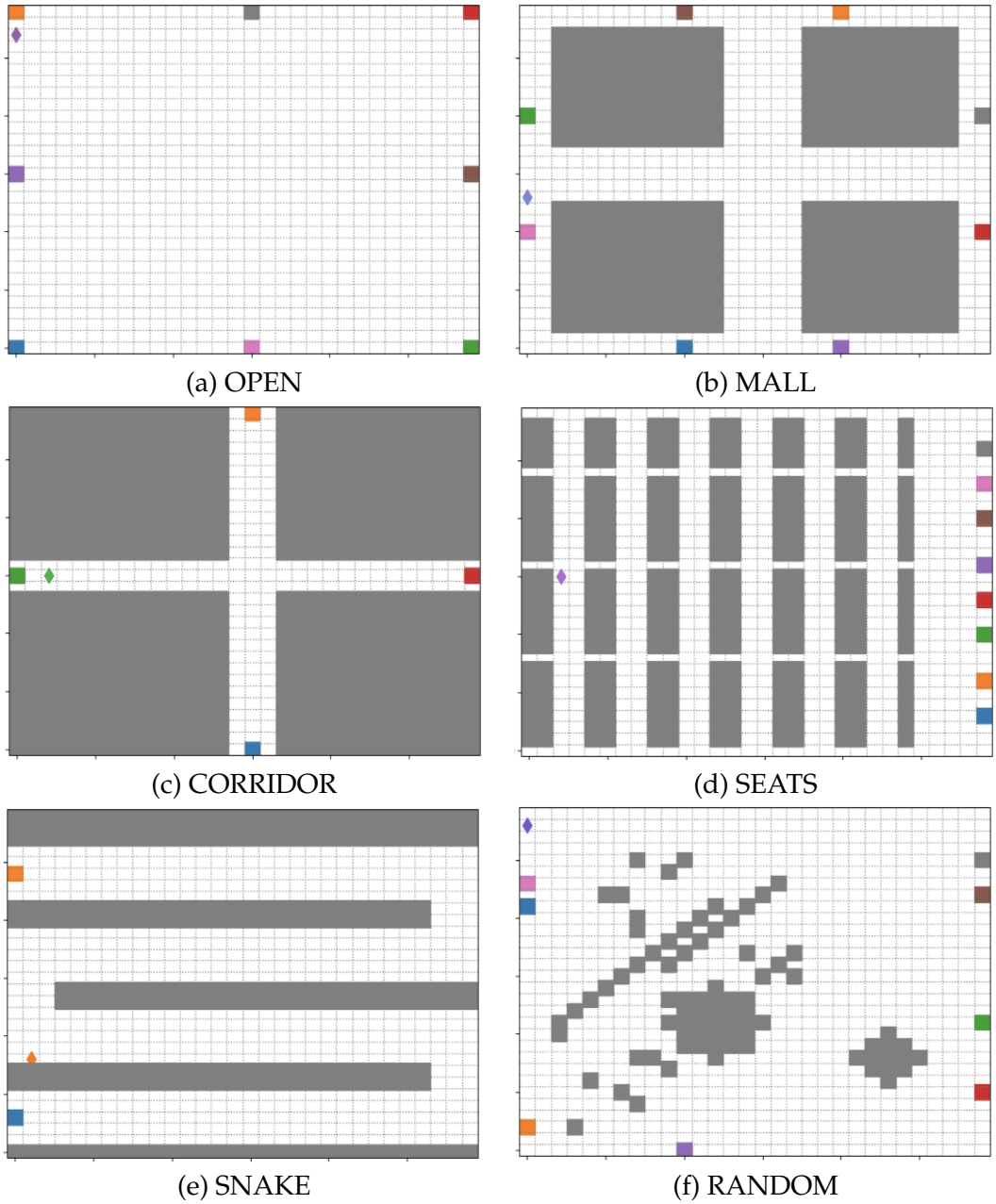
5.2.1. Entorno

En nuestro caso, el entorno será una malla bidimensional que representa los espacios por los que se moverán los agentes. Esta estará compuesta por agentes, obstáculos y salidas.

Esta malla define por defecto una serie de funcionalidades que facilitan la gestión del espacio y la interacción entre agentes. Usaremos una malla de tipo `OrthogonalMooreGrid`, la cual permite a los agentes moverse en las cuatro direcciones cardinales y en las cuatro diagonales. No permitiremos que más de un agente ocupe la misma celda al mismo tiempo, y haremos que los obstáculos y las salidas ocupen celdas enteras en la malla.

Se define un ancho y alto para la malla, y según el tipo de escenario seleccionado para la simulación, se generarán los obstáculos y las salidas en posiciones y formas específicas. Luego, los agentes aparecerán en el resto de celdas disponibles de forma aleatoria.

Se han implementado 6 tipos de escenarios distintos (OPEN, MALL, CORRIDOR, SEATS, SNAKE y RANDOM), cada uno con características topológicas específicas que permiten evaluar diferentes aspectos del comportamiento de multitudes, como se puede observar en la Figura 5.1.



(a) OPEN (b) MALL

(c) CORRIDOR (d) SEATS

(e) SNAKE (f) RANDOM

Figura 5.1: Escenarios implementados en el sistema. Cada uno presenta diferentes configuraciones de obstáculos y salidas para evaluar distintos aspectos del comportamiento de multitudes.

5.3 Navegación y Cálculo de Caminos

Para optimizar el rendimiento y evitar costosas búsquedas de ruta (como A* completo) por cada agente en cada paso, se utiliza un enfoque de **Mapas de Distancia Estáticos (Floor Fields)**.

Al inicio de la simulación, el entorno pre-calcula la distancia desde cada celda transitable hasta la salida más cercana utilizando un algoritmo de búsqueda en anchura (BFS). Esto genera un "mapa de calor.º gradiente donde cada celda contiene el coste de movimiento $d(c)$ hasta la meta.

$$d(c_{actual}) > d(c_{siguiente}) \tag{5.1}$$

Los agentes simplemente deben inspeccionar sus celdas vecinas y escoger aquella con el valor d más bajo, realizando un descenso de gradiente.

5.3.1. Diferenciación de Salidas

En escenarios complejos donde existen preferencias de salida (p.ej., "salir por donde entré"), el sistema genera múltiples mapas de distancia, uno por cada salida o grupo de salidas. Cada agente recibe una referencia al mapa que debe seguir, permitiendo flujos cruzados y comportamientos heterogéneos en la selección de rutas.

5.4 Lógica del Agente

La clase `CrowdAgent` encapsula la lógica de decisión. En cada paso de simulación (`step`), el agente ejecuta el siguiente ciclo:

5.4.1. 1. Probabilidad de Movimiento

No todos los agentes se mueven en cada tick. La decisión de intentar moverse depende de dos factores:

- **Velocidad Base**: Un valor intrínseco (p.ej., 0.8 para lentos, 1.0 para normales).
- **Factor de Multitud (Crowd Slowdown)**: Si el agente está rodeado por muchos vecinos (densidad alta), su probabilidad de movimiento disminuye, simulando la fricción física y la necesidad de ajustar el paso en aglomeraciones.

5.4.2. 2. Resolución de Conflictos y Prioridad

Para evitar el .efecto tablero de ajedrez"donde los agentes se bloquean mutuamente en ciclos, se implementa un sistema de prioridades:

- Los agentes **Agresivos** se procesan primero en la lista del Scheduler. Si ven un hueco libre que reduce su distancia a la salida, lo ocupan inmediatamente.
- Los agentes **Polite** verifican si la celda objetivo está siendo disputada. Si detectan un conflicto potencial, tienen una probabilidad de çeder el paso"(esperar) para evitar colisiones.

5.4.3. 3. Gestión de Bloqueos (Deadlock)

Se ha implementado un mecanismo de **Contador de Bloqueo** (C_{dl}) para situaciones donde el flujo se detiene (p.ej., dos flujos opuestos en un pasillo estrecho).

- Si un agente intenta moverse pero no puede, incrementa su contador C_{dl}.
- Cuando C_{dl} supera un umbral, el agente entra en estado de "Pánico/Presión".
- En este estado, el agente relaja sus reglas de movimiento: puede aceptar moverse a celdas que *no* reducen la distancia óptima (movimiento lateral) o volverse temporalmente agresivo para forzar un hueco rejilla

- El estado de bloqueo se çontagia"localmente: los agentes perciben la frustración de sus vecinos, incrementando sus propios contadores, lo que cataliza una respuesta de grupo para desatascar la zona.

CAPÍTULO 6
Resultados Experimentales

Los experimentos realizados arrojan conclusiones contraintuitivas respecto a la literatura clásica de "Faster-is-Slower". A continuación se detallan los hallazgos principales derivados de las simulaciones en los escenarios Corridor, Mall y Seats.

6.1 Impacto de la Agresividad

Contrario a la hipótesis inicial de que la competición excesiva ralentiza al grupo ("Faster-is-Slower"), nuestros resultados indican que **una mayor proporción de agentes agresivos no empeora el tiempo total de evacuación**. De hecho, en escenarios de alta densidad, la presencia de agentes agresivos tiende a mejorar ligeramente el flujo global.

- **Mecanismo de Desatasco**: Los agentes agresivos actúan como rompehielos". Al no titubear y ocupar agresivamente cualquier hueco libre, evitan que se formen estancamientos indecisos. Al salir rápidamente del sistema, liberan espacio para los agentes educados que vienen detrás.
- **Orden de Salida**: Consistentemente, los agentes agresivos son los primeros en evacuar, seguidos por los educados.

6.2 Impacto de los Agentes Lentos

La presencia de agentes lentos tiene un impacto desproporcionadamente negativo en la eficiencia del sistema. No solo tardan más en salir individualmente, sino que generan un efecto de .ºnda de choque"hacia atrás, obligando a los agentes rápidos (agresivos o educados) a reducir su velocidad efectiva al quedar atrapados detrás de ellos en pasillos estrechos o cuellos de botella.

6.3 La Ausencia del Efecto "Faster-is-Slower"

No se ha observado el fenómeno clásico donde el empuje excesivo reduce el flujo (bloqueo por arco). Esto sugiere que, en modelos discretos tipo grid sin física de fuerzas de contacto realistas, la .ªgresividad"(prioridad de turno) es computacionalmente más eficiente que la çortesía"(espera y cesión de turno), ya que maximiza la utilización del espacio-tiempo disponible.

CAPÍTULO 7
Conclusiones y Trabajo Futuro

7.1 Conclusiones

El presente trabajo ha demostrado la viabilidad de utilizar simulaciones basadas en agentes sobre rejillas discretas (MESA) para el estudio de dinámicas de evacuación, ofreciendo un equilibrio entre complejidad computacional y riqueza de comportamiento.

De los experimentos realizados, se extraen las siguientes conclusiones principales:

1. **Eficiencia de la Agresividad**: En entornos discretos, los agentes que priorizan su propio movimiento (agresivos) tienden a evacuar más rápido y, paradójicamente, pueden acelerar el flujo global al resolver situaciones de bloqueo más rápidamente que los agentes cooperativos excesivamente cautos.

2. **Distribución Espacial**: Los agentes demuestran una capacidad emergente para distribuir el flujo en pasillos paralelos. Al buscar siempre celdas vacías que minimicen la distancia, se observa un "balanceo de carga"natural entre rutas alternativas, siempre que no existan cuellos de botella extremos.

3. **Limitaciones del Modelo Discreto**: Se observa que en situaciones de flujos opuestos (contraflujo) o convergencia en ángulos rectos, los agentes tienden a bloquearse (deadlock) con mayor facilidad que en modelos de fuerzas continuas, debido a la rigidez de la rejilla y la falta de "deslizamiento"físico.

7.2 Trabajo Futuro

Para superar las limitaciones identificadas, se proponen las siguientes líneas de investigación:

- **Sistema Anti-Deadlock Avanzado**: Implementar un sistema de reputación de celda"donde, si un agente queda bloqueado mucho tiempo en una posición, marque esa celda como çostosa.en el mapa de navegación global, forzando a otros agentes a recalcular rutas (repathing) lejos de la congestión.

- **Sub-Grid Movement**: Permitir coordenadas continuas dentro de las celdas discretas para suavizar el movimiento y permitir adelantamientos más realistas.

Bibliografía

[1] Society of Fire Protection Engineers. *SFPE Handbook of Fire Protection Engineering*. 5th. Reference for Performance-Based Design methods. Springer, 2016.

[2] John J. Fruin. *Pedestrian Planning and Design*. Original Level of Service framework for pedestrian flow. Metropolitan Association of Urban Designers y Environmental Planners, 1971.

[3] ISO. *ISO 20414:2020 - Fire safety engineering — Verification and validation protocol*. Inf. téc. International Organization for Standardization, 2020.

[4] Richard D. Peacock y Paul A. Reneke. *NIST Technical Note on Evacuation Movement*. Inf. téc. Demographic and movement data for evacuation modeling. National Institute of Standards y Technology, 2008.

[5] Dirk Helbing y Péter Molnár. «Social force model for pedestrian dynamics». En: *Physical Review E* 51 (1995). Original Social Force Model publication, págs. 4282-4286.

[6] Jur van den Berg et al. «Reciprocal n-Body Collision Avoidance». En: *Robotics Research* (2011). ORCA collision avoidance algorithm, págs. 3-19.

[7] Andreas Schadschneider et al. «Evacuation Dynamics: Empirical Results, Modeling and Applications». En: *Encyclopedia of Complexity and Systems Science* (2009). Floor field models and cellular automata for crowd simulation, págs. 3142-3176.