



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



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

Simulación de Evacuación 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 en situaciones de emergencia. Utilizando el framework MESA en Python, se ha creado un modelo discreto basado en malla que permite analizar el impacto de comportamientos individuales en procesos de evacuación 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; Evacuación; Simulación de evacuación; Multitudes.

Índice general

Índice general	IV
1 Introducción	1
1.1 Alcance y Objetivos	1
1.1.1 Enfoque en Evacuación de Emergencia	2
1.2 Uso de MESA	2
1.3 Experimentación	2
1.4 Estructura del Documento	3
2 Estado del Arte	5
2.1 Objetivos de la Simulación de Evacuación	5
2.1.1 Diseño Basado en Prestaciones (PBD) y Seguridad	5
2.1.2 Nota sobre Nivel de Servicio (LOS)	6
2.2 Marcos Regulatorios y Validación	6
2.3 Arquitecturas de Navegación y Micro-comportamientos	6
2.3.1 Modelos de Espacio Continuo	6
2.3.2 Modelos Discretos y Grid-Based	6
3 Análisis del problema	7
3.1 Modelo: Discrete Grid-Based	7
3.2 Agentes: Comportamiento en Evacuación	7
3.3 Movimiento	8
3.3.1 Dead locks	8
3.3.2 Preferencias	9
3.3.3 Altas densidades	9
4 Evaluación	11
4.1 Métricas de Evaluación	11
4.2 Diseño de Experimentos	12
4.2.1 Escenarios	12
4.2.2 Variables Experimentales	12
4.3 Deadlocks	14
4.4 Nota sobre Flujos Continuos	14
5 Implementación	15
5.1 Visualización y Recolección de Datos	15
5.1.1 Visualización de Agentes y Entorno	15
5.1.2 Visualización de Gráficas y Datos	16
5.2 Estructura del Modelo	18
5.2.1 Entorno	18
5.3 Navegación y Cálculo de Caminos	18
5.3.1 Heurística en A*	19
5.3.2 Diferenciación de Salidas	19
5.4 Lógica del Agente	19
5.4.1 1. Probabilidad de Movimiento	19
5.4.2 2. Resolución Prioridad	19
5.4.3 3. Gestión de Bloqueos (Deadlock)	19

5.5	Flujo continuo	20
6	Resultados Experimentales	21
6.1	Impacto de la Agresividad	21
6.2	Impacto de los Agentes Lentos	21
6.3	La Ausencia del Efecto "Faster-is-Slower"	21
7	Conclusiones y Trabajo Futuro	23
7.1	Conclusiones	23
7.2	Trabajo Futuro	23

CAPÍTULO 1

Introducción

La gestión de evacuaciones y la seguridad en situaciones de emergencia son preocupaciones crecientes en la planificación urbana y arquitectónica. Entender cómo las personas evacuan espacios durante emergencias, cómo reaccionan bajo presión y cómo interactúan entre sí en procesos de evacuación 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 evacuación de 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 multi-agente (MAS) flexible utilizando la librería MESA en Python, que permita estudiar el impacto de los comportamientos individuales heterogéneos en la dinámica de evacuación de grupos de personas.

- **Abstracción frente a Precisión Física:** Este proyecto **no pretende** crear una herramienta de ingeniería civil certificable bajo la norma ISO 20414 (estándar del que se habla más tarde) para validación legal de planos de evacuación, ni replicar con exactitud milimétrica unidades físicas reales.
- **Enfoque en Tendencias y Comportamiento de Evacuación:** El objetivo es simular tendencias emergentes y relaciones causales en procesos de evacuación (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. Enfoque en Evacuación de Emergencia

El sistema se centra exclusivamente en el estudio de situaciones de evacuación de emergencia, donde la rapidez y eficiencia del proceso de evacuación son críticas para la supervivencia. Se pretende analizar cómo diferentes configuraciones de agentes y escenarios de evacuación afectan métricas clave como el tiempo total de evacuación, la tasa de flujo hacia las salidas y la congestión durante la evacuación.

Estos sistemas de evacuación son relevantes en contextos como:

- **Eventos Masivos:** Evacuación de conciertos, festivales y eventos deportivos en situaciones de emergencia.
- **Infraestructuras Públicas:** Planes de evacuación para aeropuertos, estaciones de tren y centros comerciales ante emergencias.
- **Diseño Urbano:** Planificación de rutas de evacuación en espacios públicos y minimización de riesgos durante evacuaciones.

Aunque el sistema también tiene la capacidad técnica de simular **flujos continuos de personas** en situaciones no urgentes, este aspecto no se ha explorado en profundidad en este trabajo. El enfoque principal es la evacuación de emergencia.

El enfoque principal y único de este trabajo es la evacuación de emergencia, validando si un espacio permite una evacuación segura y eficiente bajo diferentes condiciones y composiciones de agentes durante situaciones críticas. El sistema permite modelar de forma sencilla distintos escenarios de evacuación y composiciones de agentes en emergencias.

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 [1] y adaptando sus componentes a las necesidades específicas del estudio de evacuación.

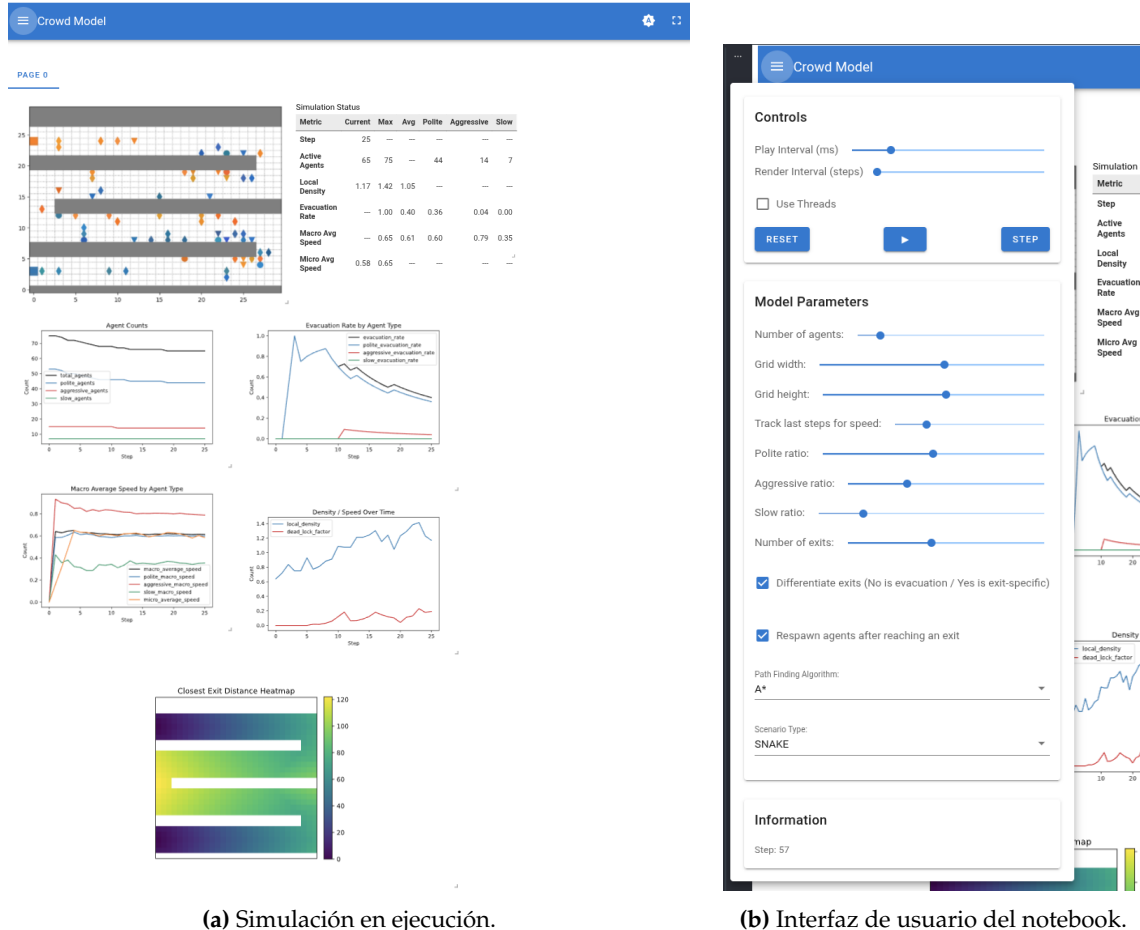
1.3 Experimentación

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 evacuación ha evolucionado de ser una disciplina académica minoritaria, a convertirse en una herramienta crítica para garantizar la seguridad en situaciones de emergencia en infraestructuras modernas.

Este capítulo evalúa el estado del arte actual en simulación de evacuación, dividiendo el análisis en los objetivos que persiguen estos sistemas: los marcos regulatorios de seguridad en evacuaciones y las arquitecturas de micro-comportamiento utilizadas para modelar la navegación de agentes durante evacuaciones.

2.1 Objetivos de la Simulación de Evacuación

En el panorama actual de seguridad en edificaciones, la simulación de evacuación responde principalmente a regulaciones de seguridad y Diseño Basado en Prestaciones para garantizar evacuaciones efectivas en situaciones de emergencia.

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 [2].

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:

$$ASET > RSET + Margen_de_Seguridad \quad (2.1)$$

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.
- **Tiempo de Viaje:** La fase física del movimiento desde el origen hasta una zona segura. Este es el componente principal que modela el sistema propuesto.

2.1.2. Nota sobre Nivel de Servicio (LOS)

Aunque existen metodologías como el Nivel de Servicio (LOS) para evaluar el confort peatonal en situaciones no urgentes [3], este trabajo se centra exclusivamente en evacuación de emergencia. Por tanto, las métricas de confort no son consideradas.

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 [4].
- **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 [5].

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 [6], trata a los agentes como partículas sometidas a fuerzas newtonianas. Combina una fuerza impulsora hacia la meta con fuerzas repulsivas para mantener la distancia personal.
- **Optimal Reciprocal Collision Avoidance (ORCA):** Opera en el espacio de velocidades para garantizar matemáticamente trayectorias libres de colisiones [7]. Aunque eficiente, puede resultar en comportamientos '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 [8].

Ya que este es el elegido para el sistema, se profundiza más adelante.

CAPÍTULO 3

Análisis del problema

El desarrollo de simuladores de evacuación implica equilibrar el realismo físico con la complejidad computacional. Dado el objetivo de estudiar comportamientos emergentes durante evacuaciones de grandes grupos, se requiere una aproximación que permita simular cientos de personas evacuando en tiempo real o cercano al tiempo real, por lo que se prioriza la ejecución 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 un agente, una pared (obstáculo) o una salida. Esto simplifica la detección de colisiones y elimina los costosos cálculos de geometrías complejas realistas.
2. **Tiempo Discreto:** La simulación avanza en pasos discretos o "steps". La velocidad de los agentes se modela probabilísticamente: un agente con "mayor velocidad" simplemente tiene una mayor probabilidad de moverse en cada step.
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 en Evacuación

Si nos fijamos en análisis reales de evacuaciones, las personas exhiben comportamientos heterogéneos bajo situaciones de emergencia. Para obtener resultados significativos sobre evacuación, el sistema debe ser capaz de modelar distintos perfiles de comportamiento de evacuación que interactúen entre sí.

En concreto, para este trabajo se han definido tres tipos de agentes con comportamientos diferenciados durante evacuaciones. Estos pretenden agrupar las características más relevantes observadas en emergencias:

- **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.

La lógica e interacción de estos agentes será muy similar, 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: el suelo o las celda, tendrán un valor asociado que indica la distancia a la salida más cercana.

Se menciona esto porque es un aspecto crucial del sistema, ya que los 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.

Además, se pretende que los agentes resuelvan estas situaciones con una serie de reglas y comportamientos adicionales. Por esto introducimos los siguientes conceptos:

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 poder moverse, la celda de algún agente 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. Lo buscamos provocar es que los agentes dejen de ir en la dirección más próxima a la salidas, y se

muevan de forma 'asustada' a celdas vecinas para intentar aliviar la congestión y / o encontrar una ruta alternativa.

También se implementa un mecanismo de influencia entre agentes: si un agente está bloqueado y sus vecinos también lo están, su contador de bloqueo aumentará más rápido. Esto simula la propagación del pánico o la frustración en situaciones de congestión.

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'.

Si una celda mejor o igual a la actual está libre, el agente se moverá a esa celda. En caso de empate, se moverá a la primera celda escaneada simulando una dirección preferida.

Salidas preferidas

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 (Evacuation 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 (Local density):** 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.

Esta métrica se usa para gestionar el comportamiento de los agentes en situaciones de congestión y alterar su lógica de movimiento.

- **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 4 ticks.

Luego, también se recogen métricas de cada una por cada tipo de agente (agresivos, educados y lentos) 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 4.1, 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 en 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 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).

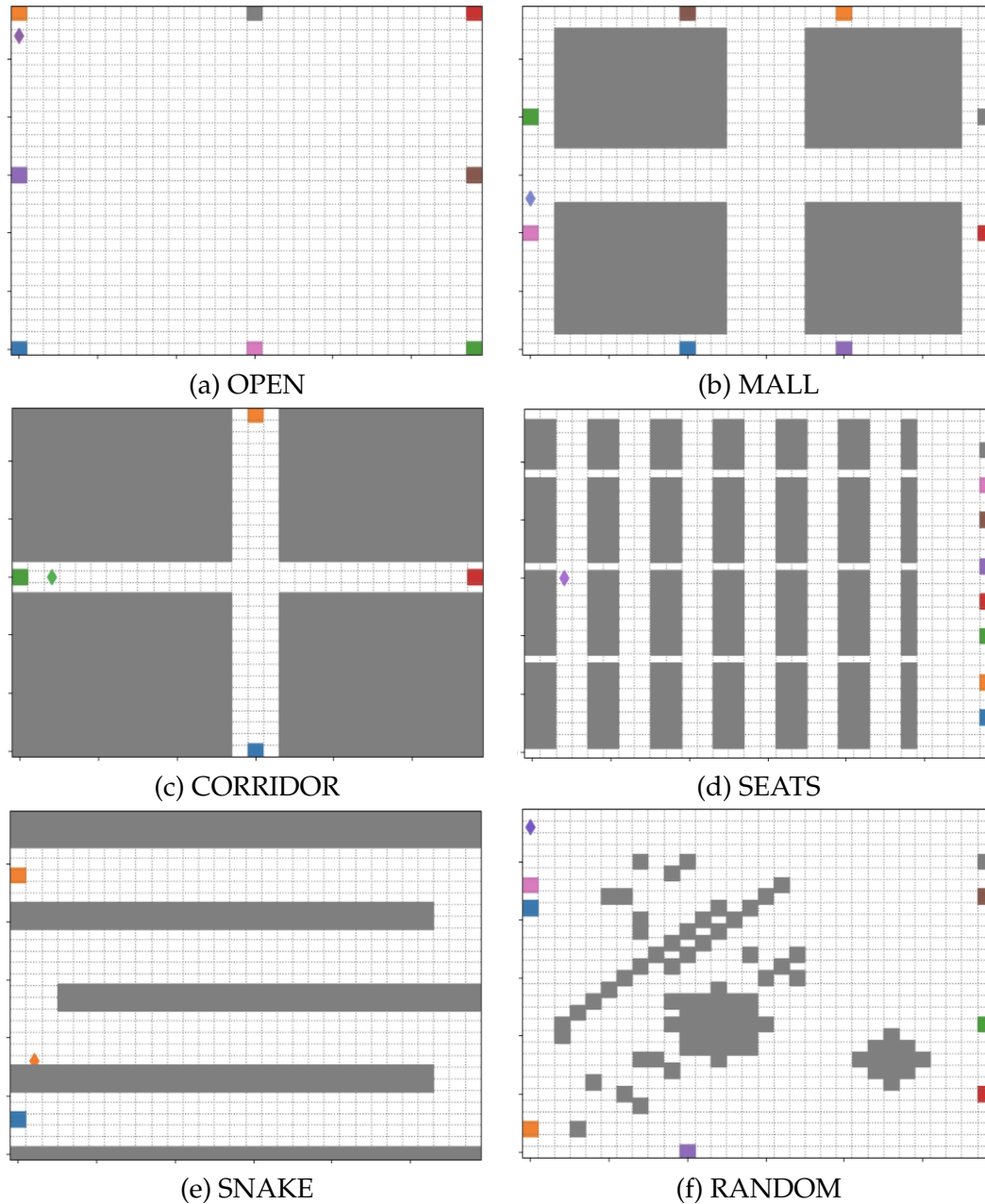


Figura 4.1: Escenarios implementados en el sistema. Cada uno presenta diferentes configuraciones de obstáculos y salidas para evaluar distintos aspectos del comportamiento de multitudes.

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 Nota sobre Flujos Continuos

Aunque el sistema tiene capacidades técnicas para simular flujos continuos de personas en situaciones no urgentes, este aspecto queda completamente fuera del alcance del presente trabajo.

Este trabajo se centra exclusivamente en escenarios de evacuación de emergencia donde el objetivo es vaciar completamente el espacio, no en situaciones de flujo diario o confort peatonal.

Los flujos continuos podrían estudiarse en trabajos futuros activando la opción de *reaparición de agentes*, pero no se consideran relevantes para el estudio de evacuación que nos ocupa.

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.

Se han implementado visualizadores personalizadas, en las que hemos recolectado los datos de la simulación y se muestran en gráficos con tablas en tiempo real.

5.1.1. Visualización de Agentes y Entorno

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 agentes tengan una salida favorita, se usarán los colores de cada una en lugar de los colores por defecto.

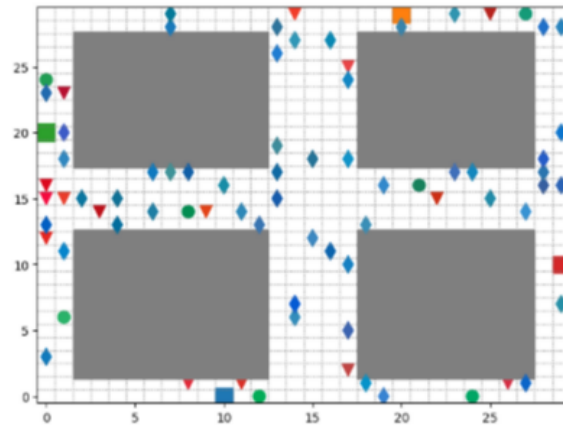


Figura 5.1: Visualización de los diferentes tipos de agentes y elementos del entorno con sus respectivas formas y colores.

Esto se ha hecho a partir de la definición de una función de `portrayal` personalizada que asigna colores y formas según el tipo de agente o elemento del entorno.

5.1.2. Visualización de Gráficas y Datos

Se han creado gráficas en tiempo real de las métricas, 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.

La Figura 5.2 muestra las principales métricas recolectadas durante una simulación, incluyendo el conteo de agentes a lo largo del tiempo, las tasas de evacuación por tipo de agente, la velocidad macro promedio y la evolución de la densidad y el factor de bloqueo. Estas gráficas permiten observar en tiempo real el comportamiento emergente del sistema y identificar patrones como cuellos de botella o situaciones de deadlock.

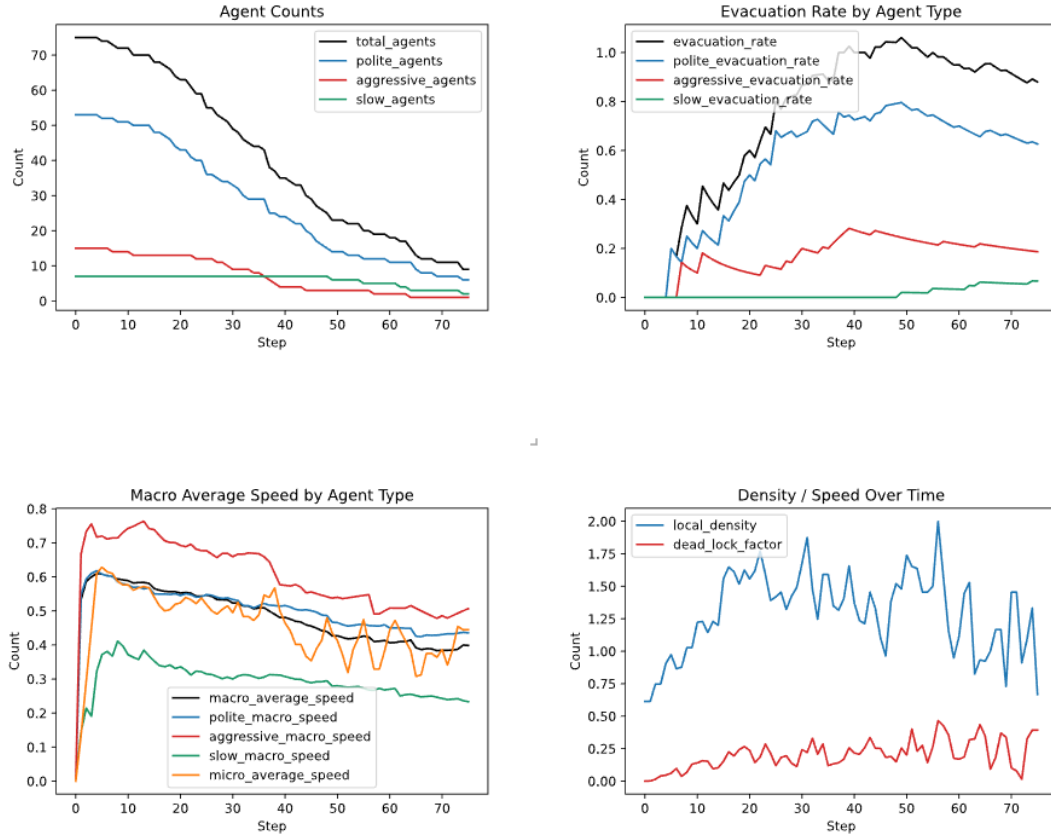
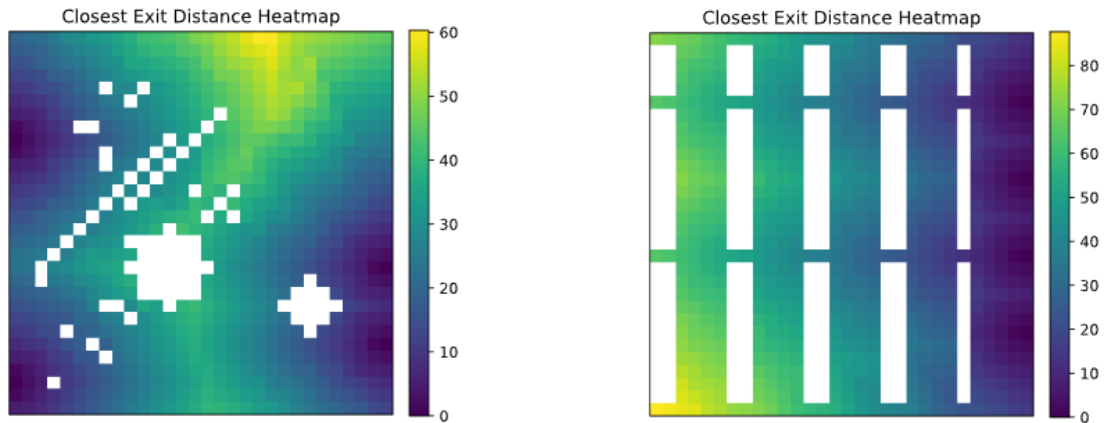


Figura 5.2: Visualización de métricas en tiempo real durante una simulación de evacuación. Se muestran cuatro gráficas principales: conteo de agentes por tipo, tasas de evacuación diferenciadas, velocidades macro promedio por tipo de agente, y evolución de la densidad local junto con el factor de bloqueo.

Adicionalmente, el sistema genera mapas de calor que visualizan la distancia desde cada celda hasta la salida más cercana, como se muestra en las Figuras 5.3a y 5.3b. Estos mapas de distancia son fundamentales para la navegación de los agentes, ya que constituyen los *floor fields* que guían su movimiento hacia las salidas durante la evacuación.



(a) Mapa de distancias en escenario RANDOM con obstáculos aleatorios.

(b) Mapa de distancias en escenario SEATS tipo auditorio.

Figura 5.3: Mapas de calor mostrando la distancia a la salida más cercana en diferentes escenarios. Los colores más claros (amarillo) indican mayor distancia, mientras que los más oscuros (púrpura) indican proximidad a las salidas. Las zonas blancas representan obstáculos o salidas.

En la Figura 1.1 se pueden ver capturas de pantalla del sistema completo de simulación 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 adaptándose al tamaño definido. 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 4.1.

5.3 Navegación y Cálculo de Caminos

Para optimizar el rendimiento y evitar costosas búsquedas de ruta 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) o uno con heurística A*. Esto genera un 'mapa de calor' o gradiente donde cada celda contiene el coste de movimiento $d(c)$ hasta la meta.

$$d(c_{actual}) \geq d(c_{siguiente}) \quad (5.1)$$

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

No se requiere que los agentes recalculen rutas dinámicamente, lo que reduce significativamente la carga computacional y permite simular grandes multitudes en tiempo real. Y mencionar que se hace \geq y no solo $>$ para permitir movimientos laterales en caso de deadlocks.

5.3.1. Heurística en A*

Ya que el movimiento de los agentes permite movimientos diagonales, la heurística utilizada en A* es la **Distancia Euclidiana**, que calcula el coste estimado con una pequeña optimización para evitar agentes atascados en esquinas:

$$H(current, goal) = 0,5 \times \sqrt{(x_{goal} - x_{current})^2 + (y_{goal} - y_{current})^2}$$

Ese factor de 0.5 se ha añadido para reducir la influencia de la heurística y favorecer caminos más directos en lugar de diagonales excesivas que podrían llevar a bloqueos. Se ha llegado al valor tras experimentación.

5.3.2. 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. Cada agente recibe una referencia al mapa que debe seguir, permitiendo flujos cruzados y comportamientos heterogéneos en la selección de rutas.

En caso de ir a la más cercana, el agente recibe todos los mapas y evalúa el mínimo en cada paso.

5.4 Lógica del Agente

La clase `CrowdAgent` descende de la clase `mesa.Agent` y 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 aleatorio que se elige para cada agente entre un rango concreto para cada tipo de agente. Este valor representa la probabilidad base de que el agente intente moverse en cada tick. Los agentes *Slow* tendrán una probabilidad base menor que los *Polite* y estos a su vez menor que los *Aggressive*.
- **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. Este efecto es más pronunciado en agentes *Polite* y menos en *Aggressive* y en los *Slow*.

5.4.2. 2. Resolución Prioridad

Para crear la ilusión de diferentes niveles de agresividad, se implementa una lógica simple en la que, para cada `step`, primero los agentes *Aggressive* se moverán a las celdas que deseen ocupar, seguidos por los *Polite* y los *Slow* en orden aleatorio.

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 y puede incluso retroceder o moverse lateralmente para aliviar la congestión, dejando pasar a otros agentes.
- El estado de bloqueo se contagia "localmente": los agentes perciben la frustración de sus vecinos, incrementando sus propios contadores, lo que cataliza una respuesta de grupo para desatascar la zona.

Cuando la congestión se resuelve, el contador se reduce rápidamente, permitiendo que el agente vuelva a su comportamiento normal.

5.5 Flujo continuo

Como se ha mencionado, el sistema tiene la capacidad técnica de simular flujos continuos de personas en situaciones no urgentes, pero este aspecto queda completamente fuera del alcance del presente trabajo.

Únicamente mencionar que su implementación es sencilla: se activa la opción de *reaparición de agentes* en el modelo, de forma que cuando un agente evacua, se crea uno nuevo en una posición aleatoria del entorno.

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 "cortesía" (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 procesos de evacuación en emergencias, ofreciendo un equilibrio entre complejidad computacional y riqueza de comportamiento en situaciones de evacuación.

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

1. **Eficiencia de la Agresividad en Evacuación:** En procesos de evacuación con entornos discretos, los agentes que priorizan su propia evacuación (agresivos) tienden a evacuar más rápido y, paradójicamente, pueden acelerar la evacuación global al resolver situaciones de bloqueo más rápidamente que los agentes cooperativos excesivamente cautos durante la evacuación.
2. **Distribución Espacial en Evacuación:** Los agentes demuestran una capacidad emergente para distribuir el flujo de evacuación en pasillos paralelos. Al buscar siempre rutas hacia las salidas que minimicen la distancia, se observa un "balanceo de carga" natural entre rutas de evacuación alternativas, siempre que no existan cuellos de botella extremos.
3. **Limitaciones del Modelo Discreto en Evacuación:** Se observa que en situaciones de flujos opuestos durante la evacuación o convergencia en ángulos rectos hacia las salidas, 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 costosa.^{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] Jackie Kazil, David Masad y Andrew Crooks. *MESA: Agent-based Modeling in Python*. Python framework for agent-based modeling, 2020. URL: <https://mesa.readthedocs.io/>.
- [2] Society of Fire Protection Engineers. *SFPE Handbook of Fire Protection Engineering*. 5th. Reference for Performance-Based Design methods. Springer, 2016.
- [3] John J. Fruin. *Pedestrian Planning and Design*. Original Level of Service framework for pedestrian flow. Metropolitan Association of Urban Designers y Environmental Planners, 1971.
- [4] ISO. *ISO 20414:2020 - Fire safety engineering — Verification and validation protocol*. Inf. téc. International Organization for Standardization, 2020.
- [5] 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.
- [6] 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.
- [7] Jur van den Berg et al. «Reciprocal n-Body Collision Avoidance». En: *Robotics Research* (2011). ORCA collision avoidance algorithm, págs. 3-19.
- [8] 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.