Optimización y Robotización de Almacenes para e-commerce

Benchmark de planeación, asignación y coordinación multi-robot

1. Descripción general

Este proyecto implementa un **simulador discreto y reproducible** de una **flota de robots móviles** operando en un **Centro de Distribución (CEDIS)** típico de e-commerce.

El sistema permite:

1. Generar **layouts realistas de un almacén** (anaqueles, pasillos y estaciones).
2. Generar **pedidos discretos** (eventos en tiempo discreto), con control temporal mediante ticks.
3. Simular la **operación completa del CEDIS**, incluyendo:
    - asignación de pedidos a robots,
    - planeación de rutas,
    - coordinación multi-robot,
    - detección de bloqueos y esperas.
4. **Visualizar la simulación** mediante:
    - animación en video,
    - heatmaps de tráfico, congestión y espera.
5. Obtener **métricas cuantitativas** de desempeño del sistema.

El proyecto está diseñado explícitamente como **reto académico y benchmark experimental**, no como un sistema industrial ni como un optimizador global de almacenes.

2. Alcance y propósito

El objetivo del benchmark es **estudiar cómo distintas decisiones locales** (asignación de pedidos, planeación de rutas y mecanismos de coordinación) impactan el desempeño global del sistema.

El simulador **no persigue una solución globalmente óptima**, sino que proporciona un **caso de estudio** técnicamente consistente, controlado y reproducible, que permite:

- comparar estrategias de decisión,
- analizar trade-offs operativos,
- justificar mejoras mediante métricas y razonamiento de ingeniería.

3. Estructura del proyecto

```
sim_almacen/
├─ README.md
├─ requirements.txt
├─ a_estrella.py
├─ demo_final.py
├─ generador_layout.py
```

```
├─ generador_pedidos.py
├─ out_paths.py
├─ sim_core.py
├─ tabla_reservas.py
├─ visualiza_simulacion.py
├─ pruebas/
│  ├─ prueba_a_estrella.py
│  └─ prueba_tabla_reservas.py
└─ outputs/
   └─ <escenario>/
      ├─ anaqueles.json
      ├─ estaciones.json
      ├─ metricas.json
      ├─ pedidos.json
      ├─ spawn.json
      ├─ layout.npy
      ├─ heatmap_esperas.png
      ├─ heatmap_ratio.png
      ├─ heatmap_visitas.png
      ├─ layout.png
      └─ simulacion.mp4
```

Los archivos se organizan por tipo y se listan alfabéticamente para facilitar la navegación.

4. Convención de escenarios

Todo resultado del benchmark se agrupa bajo la carpeta:

```
outputs/<escenario>/
```

Ejemplos válidos:

```
outputs/seed42/
outputs/experimento_A/
outputs/prueba_estres/
```

El contenido de la carpeta de un escenario no incluye scripts o librerías, simplemente identifica un conjunto coherente de archivos (layout, pedidos, métricas y visualizaciones).

5. Flujo de ejecución recomendado

Paso 0 – Instalación de dependencias

Es muy probable que ya tengan instaladas las librerías requeridas: `numpy` y `matplotlib`; caso contrario, pueden instalarlas usando el comando:

```
pip install -r requirements.txt
```

Paso 1 – Generar el layout del CEDIS

```
python generador_layout.py --escenario seed42 --seed 42 --ancho 120 --alto 80 --
estaciones 20
```

Genera en `outputs/seed42/`:

- `layout.npy` → representación discreta del almacén,
- `estaciones.json`,
- `anaqueles.json`,
- `spawn.json`.

El parámetro `--seed` controla la generación pseudoaleatoria del layout; al fijarlo, se garantiza que el mismo escenario pueda regenerarse exactamente.

Paso 2 – Generar pedidos discretos

```
python generador_pedidos.py --escenario seed42 --pedidos 600 --burst
```

Genera:

- `pedidos.json`

El flag `--burst` introduce **ráfagas temporales de demanda**, simulando picos de carga sobre el sistema.

Paso 3 – Ejecutar la simulación (benchmark)

```
python demo_final.py --escenario seed42 --robots 20 --ticks 10000
```

Produce:

- `metricas.json`,
- métricas impresas en consola, tales como:
  - pedidos completados,
  - throughput,
  - utilización promedio de robots,
  - eventos de bloqueo,
  - distancia recorrida.

Paso 4 – Visualizar la simulación

Para generar el video, emplearemos FFMPEG (https://www.ffmpeg.org/) porque es una alternativa más robusta que Pillow; es necesario descargar e instalar FFMPEG para posteriormente agregarlo al path para

poder emplearlo.

```
python visualiza_simulacion.py --escenario seed42
```

Genera:

- `layout.png`,
- `simulacion.mp4`,
- `heatmap_visitas.png`,
- `heatmap_esperas.png`,
- `heatmap_ratio.png`.

6. Componentes principales del sistema

`sim_core.py`

Núcleo del simulador:

- modelo de robots, pedidos y estados,
- asignación de pedidos mediante decisiones locales inmediatas,
- avance por ticks discretos,
- coordinación mediante **tabla de reservas**,
- cálculo de métricas globales.

`a_estrella.py`

Implementación del algoritmo **A*** para planeación de rutas sobre una retícula.

`tabla_reservas.py`

Mecanismo de coordinación temporal que evita:

- colisiones de vértice,
- intercambios de arista,
- inconsistencias espacio-tiempo.

No decide rutas ni asignaciones: **solo aplica exclusión y seguridad**.

`generador_layout.py`

Genera layouts tipo CEDIS con:

- anaqueles,
- pasillos principales,
- cross-aisles,
- estaciones,
- puntos de spawn.

Switches disponibles

Switch	Tipo	Default	Descripción
`--escenario`	`str`	`"seed42"`	Nombre lógico del escenario. Determina la carpeta `outputs/<escenario>/` donde se escriben todos los artefactos generados.
`--seed`	`int`	`42`	Semilla pseudoaleatoria utilizada para generar el layout. Controla completamente la reproducibilidad del escenario.
`--ancho`	`int`	`120`	Ancho del grid discreto que representa el CEDIS (número de columnas).
`--alto`	`int`	`80`	Alto del grid discreto que representa el CEDIS (número de filas).
`--estaciones`	`int`	`20`	Número de estaciones de consolidado de pedidos. Se colocan en el borde sur del layout.

`generador_pedidos.py`

Genera pedidos discretos con:

- distribución uniforme o en ráfaga,
- control determinista por seed.

Switches disponibles

Switch	Tipo	Default	Descripción
`--escenario`	`str`	`"seed42"`	Nombre del escenario. Determina la carpeta `outputs/<escenario>/` desde la cual se leen los archivos de layout (`estaciones.json`, `anaqueles.json`) y donde se escribe la salida.
`--seed`	`int`	`42`	Semilla pseudoaleatoria utilizada para generar los pedidos sintéticos. Afecta la selección de anaqueles, estaciones y ticks de creación.
`--pedidos`	`int`	`600`	Número total de pedidos discretos a generar. Cada pedido corresponde a un evento independiente en tiempo discreto.
`--burst`	`flag`	`False`	Si se activa, genera un patrón de demanda en ráfaga: el 70% de los pedidos se crean en los ticks iniciales (0–2000) y el resto se distribuye hasta el tick 10000. Si no se activa, todos los pedidos se crean en el tick 0.

`demo_final.py`

Ejecuta el benchmark completo del CEDIS, coordinando:

- la carga del layout y los pedidos del escenario,
- la simulación multi-robot por ticks discretos,
- el cálculo y almacenamiento de métricas de desempeño.

Este script no define la estructura del sistema; consume exclusivamente los artefactos generados previamente para un escenario dado.

Switches disponibles

Switch	Tipo	Default	Descripción
`--escenario`	`str`	`"seed42"`	Nombre del escenario. Lee los archivos de entrada desde `outputs/<escenario>/` y escribe ahí las métricas generadas.
`--seed`	`int`	`42`	Semilla pseudoaleatoria utilizada por el simulador para inicializar decisiones internas reproducibles.
`--robots`	`int`	`20`	Número de robots activos en la simulación. Define el tamaño de la flota.
`--ticks`	`int`	`10000`	Número total de ticks discretos que dura la simulación.

`visualiza_simulacion.py`

Herramientas de análisis visual:

- animación del sistema,
- estados de robots por color,
- heatmaps de congestión y espera.

Nota: Actualmente invoca FFMPEG mediante un path fijo para asegurar funcionamiento consistente en los entornos objetivo. A futuro se resolverá la correcta implementación con la bandera --ffmpeg_path

Switches disponibles

Switch	Tipo	Default	Descripción
`--escenario`	`str`	`"seed42"`	Nombre del escenario. Lee entradas desde `outputs/<escenario>/` y escribe visualizaciones en la misma carpeta.
`--seed`	`int`	`42`	Semilla utilizada para inicializar el estado interno del simulador durante la visualización.
`--robots`	`int`	`20`	Número de robots a visualizar en la simulación.
`--ticks`	`int`	`10000`	Número total de ticks discretos a simular durante la animación.
`--pasos_por_frame`	`int`	`25`	Número de ticks simulados por cada frame del video. Controla la velocidad visual de la animación.
`--fps`	`int`	`20`	Frames por segundo del video generado.

`out_paths.py`

Define y valida la estructura de carpetas de salida:

- creación automática de `outputs/<escenario>/`,
- normalización de rutas para métricas y visualizaciones.

7. Reproducibilidad

El sistema es determinista al fijar el `--seed` durante la generación del layout y al reutilizar los artefactos asociados a un mismo `--escenario`.

Esto permite:

- repetir experimentos,
- comparar estrategias,
- evaluar mejoras de forma justa.

8. Uso académico

Este benchmark está diseñado como **reto integrador** para equipos multidisciplinarios:

- **IRS**: navegación, abstracción robótica, interacción humano-robot.
- **ITC**: simulación, algoritmos, estructuras de decisión.
- **ITD**: datos, métricas, análisis y reproducibilidad.

Los estudiantes pueden:

- modificar heurísticas,
- cambiar políticas de asignación,
- introducir fallas o restricciones,
- medir su impacto en métricas globales.

9. Filosofía de diseño

Este es un proyecto vivo. Aunque el benchmark es funcional y permite ejecutar experimentos reproducibles, no se considera una implementación final ni pulida, sino una base en evolución, diseñada para ser extendida, refinada y mejorada a lo largo del tiempo.

El sistema separa deliberadamente **políticas** de **mecanismos**.

Esto da lugar a un **caso de estudio deliberadamente subóptimo**, pero técnicamente consistente, diseñado para:

- realizar análisis comparativos entre estrategias,
- discutir trade-offs operativos de manera informada,
- proponer mejoras sustentadas en métricas y razonamiento de ingeniería.

10. Glosario

Asignación de pedidos: Proceso mediante el cual un pedido se asigna a un robot disponible. En la implementación de referencia, esta asignación se realiza mediante decisiones locales inmediatas, sin optimización global.

Benchmark: Implementación de referencia diseñada para comparar estrategias de decisión bajo condiciones controladas y reproducibles. No representa una solución óptima ni un sistema industrial.

Burst (ráfaga de pedidos): Patrón de generación de pedidos en el que múltiples eventos discretos se concentran en un intervalo corto de ticks, produciendo picos temporales de carga sobre el sistema.

CEDIS (Centro de Distribución): Abstracción de un almacén de e-commerce donde una flota de robots transporta anaqueles hacia estaciones de consolidación.

Coordinación multi-robot: Conjunto de mecanismos que permiten que múltiples robots compartan el espacio sin colisiones ni inconsistencias temporales. En este benchmark se limita a exclusión y sincronización, no a optimización global.

Deadlock: Situación operacional en la que, durante un tick de simulación, ningún robot logra avanzar de celda a pesar de que al menos uno se encuentra en estado activo. Es un evento de estancamiento temporal asociado a congestión espacial, conflictos de reservas o decisiones locales de movimiento. Se contabiliza como una métrica de fricción del sistema y puede resolverse en ticks posteriores sin intervención externa.

Escenario: Conjunto coherente de archivos de entrada y salida asociados a una ejecución del benchmark. Las carpetas de escenario no contienen scripts ni librerías; únicamente agrupan layouts, pedidos, métricas y visualizaciones compatibles entre sí.

Evento discreto: Ocurrencia definida en un instante de tiempo discreto (tick). En este benchmark, los pedidos se modelan como eventos discretos.

Implementación de referencia: Conjunto de políticas y mecanismos incluidos en el benchmark base, técnicamente consistentes pero deliberadamente no optimizados, utilizados como punto de comparación experimental.

Layout: Representación discreta del almacén, modelada como una retícula, que define la ubicación de anaqueles, pasillos, estaciones y puntos de spawn.

Lead time (tiempo de ciclo del pedido): Tiempo promedio transcurrido entre la creación de un pedido (tick de generación) y su finalización completa en la simulación, incluyendo espera, asignación, transporte del anaquel a la estación y retorno a su ubicación.

Métrica de desempeño: Valor cuantitativo calculado a partir de la simulación para evaluar el comportamiento del sistema (throughput, utilización, tiempos de espera, distancia recorrida, entre otros).

Pedido: Entidad discreta que representa la solicitud de transporte de un anaquel desde su ubicación hacia una estación y de regreso. No modela picking individual ni contenido del anaquel.

Planeación de rutas: Cálculo de una trayectoria válida dentro del layout desde una posición inicial hasta un objetivo.

Reproducibilidad: Propiedad del sistema que garantiza que una ejecución puede repetirse exactamente al fijar el escenario y la semilla aleatoria.

Tabla de reservas: Mecanismo de coordinación espacio-tiempo que evita colisiones y conflictos mediante la reserva anticipada de celdas en determinados ticks. No decide rutas ni asignaciones.

Throughput: Métrica de desempeño que cuantifica la capacidad del sistema para completar pedidos. En este benchmark, se define como el número total de pedidos completados durante la simulación dividido entre la duración total de la simulación en ticks.

Tick: Unidad básica de avance temporal del simulador. Todas las decisiones, movimientos y eventos ocurren en tiempo discreto.

Trade-off: Compromiso entre métricas de desempeño en el que mejorar un aspecto del sistema puede degradar otro. El análisis de trade-offs es un objetivo explícito del benchmark.

¿Por qué la semilla del benchmark es 42?

The Hitchhiker's Guide to the Galaxy