Tecnológico de Monterrey

Bloque:

Modelación de sistemas multiagentes
con gráficas computacionales (Gpo 302)

Entregable:

MA. Actividad: Roomba
Informe Técnico

Alumno:

Luis Emilio Veledíaz Flores - A01029829

Profesor:

Octavio Navarro Hinojosa

Fecha de entrega:

19 de Noviembre de 2025

1. Problema y Propuesta de Solución

a. Definición del Problema

El problema que se quiere abordar en esta actividad es la limpieza autónoma y eficiente de espacios con agentes inteligentes. Se diseñaron y simularon dos sistemas de agentes que son capaces de explorar un ambiente desconocido con obstáculos, para identificar y limpiar áreas sucias, gestionando su batería y tomando en cuenta la ubicación de cargadores (en el caso de la primera simulación es un solo cargador).

El desafío en general fue que los agentes debían operar enfrentándose a restricciones de su batería, debía moverse en grids con obstáculos, y en el caso de múltiples agentes, coordinarse y comunicarse de manera efectiva para optimizar el uso de recursos. Los agentes no contaban con un mapa previo del ambiente, por lo que debían construir el suyo a través de la exploración del grid. Para optimizar la recolección de datos y el análisis de desempeño, ambas simulaciones implementan un sistema de métricas en tiempo real que rastrea 4 indicadores: número total de movimientos, porcentaje de limpieza, límite de pasos como referencia, y batería disponible. Esto permite visualizar el comportamiento de los agentes a lo largo del tiempo y comparar su eficiencia.

b. Propuesta de Solución

Mi propuesta se separa en dos modelos de simulación, cada uno de ellos aborda de manera independiente los requerimientos de la actividad. El primero es un modelo de un agente individual, el segundo es un modelo de múltiples agentes. Ambos modelos implementan una estructura basada en una máquina de estados que permite que los agentes tomen decisiones en base a su percepción del ambiente. Mi solución incorpora varios elementos importantes:

- **Distancia Manhattan**: que permite a los agentes desplazarse eficientemente hacia sus cargadores cuando su batería está baja.
- **Memoria de tiles**: que guía la exploración del ambiente.
- **Arquitectura de subsunción**: que establece prioridades entre las actividades de los agentes.
- **Intercambio de información**: que permite que los agentes aprendan la ubicación de los otros cargadores presentes (únicamente para el modelo de múltiples agentes).

La propuesta pasó por varias versiones, donde tomé como posibilidad la distancia Euclidiana, y hacer un BFS, al final se quedó la distancia Manhattan. Esta implementación terminó

siendo una efectiva, tanto para el caso del roomba individual, como para el caso de más roombas.

2. Diseño de los Agentes

a) Objetivo General

El objetivo de cada roomba es limpiar tiles sucias que encuentre en el ambiente, mientras gestiona su porcentaje de batería. Este objetivo principal se puede dividir en dos objetivos más chiquitos:

1. Explorar el ambiente para identificar tiles sucios.
2. Mantener un nivel de batería suficiente para seguir explorando/limpiando.

El agente debe tener un balance entre estas dos metas, reconociendo que una exploración externa haciendo caso omiso a su porcentaje de energía terminaría en que se le termine la batería y termine muriendo.

b) Capacidades Efectectoras (Actuadores)

Los agentes poseen un conjunto de capacidades para actuar en el ambiente:

- **Movimiento**: Los agentes pueden desplazarse a cualquiera de sus 8 tiles adyacentes siempre y cuando no sean un obstáculo. Cada uno de estos movimientos resta 1% de batería, es una capacidad fundamental para la simulación.
- **Limpieza**: Cuando el agente se encuentra un tile sucio (distinguido por el color café), puede ejecutar una acción para limpiar, lo que convierte a esa tile en una tile limpia. Esta acción también resta 1% de batería.
- **Carga**: Cuando el agente se encuentra en una estación de carga, puede recargar su batería. Cada step en el cargador aumenta un 5% de la batería total, hasta un máximo de 100%. Esta acción en específico no consume batería, que también es un aspecto real de los roombas.

Estas capacidades de cada agente son suficientes para que ellos puedan realizar todas las tareas necesarias para cumplir el objetivo general.

c) Percepción

El sistema de percepción del agente está diseñado para acercarse lo más que se pueda a la realidad, reflejando las capacidades de un roomba real con sensores. El agente puede detectar el contenido de todas las tiles de sus vecinos inmediatos. Esto incluye la presencia de obstáculos, tiles sucios, y otras características del ambiente. No tiene una visión de largo

alcance, no se sabe el mapa desde el inicio. El agente también es capaz de conocer su posición actual en el grid mediante su coordenada (x, y). También tiene conocimiento de su nivel de batería actual, su estado en la máquina de estados y el contador de acciones que ha realizado hasta ese punto en específico.
El agente tiene un registro de todas las tiles que ha visitado, esa memoria le permite tomar decisiones de movimiento basadas en donde ha estado, lo que le permite ponerle más atención a explorar áreas nuevas.

d) Autonomía

Los agentes tienen un nivel relativamente bueno de proactividad y autonomía. No reciben órdenes externas, ni mías ni de ninguna autoridad, más bien, toman sus decisiones basándose en su percepción local y de su estado actual. La proactividad se puede notar en varios aspectos como el proceso percepción -> decisión -> acción.
Además de eso, el agente anticipa sus necesidades a futuro (p.e. reconociendo cuando su batería está cerca de agotarse) y toma acciones preventivas como consecuencia. En el modelo donde hay más de un agente, la proactividad se puede ver en la comunicación, cuando dos agentes se encuentran en alguna tile que entre en la vecindad de Moore, intercambian información sin requerir un protocolo tal cual, lo que mejora el conocimiento colectivo de todos los agentes involucrados en el grid.

e) Métricas de Desempeño

Para esta actividad defino cuatro métricas principales para evaluar el desempeño de los agentes:

- Movimiento Total: Cuenta el número total de acciones de movimiento realizadas por los agentes.
- Límite de Pasos: Una línea de referencia que muestra el porcentaje de pasos ejecutados respecto al máximo que se customiza.
- Porcentaje de Limpieza: Es el número de tiles limpias por el agente dividido entre el número total de tiles sucias al inicio de la simulación.
- Batería: En la simulación 1, es el porcentaje de batería del agente. En la simulación 2 es el promedio de la batería de todos los agentes.

3. Arquitectura de Subsunción de los Agentes

La máquina de estados de los agentes consta de cinco estados distintos, cada uno ligado a un conjunto de prioridades que garantiza la supervivencia, organizados en diferentes niveles:

- **Estado “Charging”**: Este es el estado más fundamental y tiene mayor prioridad. Cuando el agente se encuentra en un cargador, permanece en este estado hasta que su batería alcance el 80%. Una vez que alcanza este nivel, pasa al estado “Exploring”. Si el agente entra en este estado pero no se encuentra en un cargador, pasa a “MovingToCharge”.
- **Estado “MovingToCharge”**: Cuando el agente detecta que su batería está baja (por debajo del 30% o es insuficiente para regresar a su cargador), pasa a este estado. En este estado, el agente utiliza un algoritmo de distancia Manhattan para ir hacia su cargador más cercano (y que conozca). El estado bloquea todos los otros comportamientos, lo que asegura su supervivencia, y que esta sea prioridad.
- **Estado “Exploring”**: En este estado, el agente se dedica a explorar nuevas secciones del grid. Este estado es el comportamiento proactivo del agente, pero es importante recalcar que para la simulación 1, el agente usa un algoritmo BFS para encontrar la celda no visitada más cercana. En la simulación 2 se usa un enfoque más simple basado en una selección aleatoria entre vecinos no visitados.
- **Estado “MovingToDirt”**: Cuando el agente en estado “Exploring”, detecta tiles sucias en alguna de sus tiles vecinas. En este estado, el agente se mueve hacia esa tile sucia. Este estado bloquea la exploración porque sabe que limpiar es mejor que continuar explorando.
- **Estado “Cleaning”**: Cuando el agente llega a un tile sucio, pasa a este estado y ejecuta la acción de limpieza. El agente se queda en este estado hasta que se limpia la tile. Una vez limpia, regresa al estado “Exploring” a menos que su batería esté baja, en ese caso pasa a “MovingToCharge”

Las transiciones entre estados siguen las siguientes reglas de prioridad:

1. Si batería <= 0%: Estado "Charging"
2. Si está en "Charging":
    - Si está en cargador: Recargar()
    - Si batería >= 80%: "Exploring"
    - Si no está en cargador: "MovingToCharge"

3. Si está en "MovingToCharge":
    - Moverse hacia cargador
    - Si llegó al cargador: "Charging"

4. Si está en "Exploring":
    - Si hay suciedad actual: "Cleaning"
    - Si hay suciedad vecina: "MovingToDirt"
    - Si batería baja: "MovingToCharge"
    - Si no: Explorar()

5. Si está en "MovingToDirt":
    - Moverse hacia suciedad
    - Si llegó a suciedad: "Cleaning"
    - Si batería baja: "MovingToCharge"
    - Si no hay suciedad: "Exploring"

6. Si está en "Cleaning":
    - Limpiar()
    - Si batería baja: "MovingToCharge"
    - Si no: "Exploring"

4. Características del Ambiente

a) Clasificaciones del ambiente

El ambiente es inaccesible, aunque el estado completo del grid existe, cada agente individual solo puede percibir su entorno local. Un agente conoce el contenido de su tile actual y el de sus 8 vecinos, pero no tienen acceso a información sobre tiles que estén más lejos que eso.

El ambiente también es completamente no determinístico, ya que cada agente individual no puede predecir con certeza el resultado de sus acciones por factores como otros agentes, y la falta de conocimiento de las tiles fuera de rango.

Para el caso del agente individual, el ambiente es episódico. Cada simulación es un episodio independiente y completamente delimitado que comienza con un estado inicial y termina cuando se cumple la condición. El desempeño del agente en un episodio no se ve afectado por episodios anteriores o futuros. En la simulación con más agentes el ambiente es no episódico, las decisiones y el desempeño de un agente dependen de lo que hacen los otros agentes.

El ambiente es estático, la posición de obstáculos, los tiles sucios y los cargadores se establecen al inicio y ahí permanecen; en algunos casos desaparecen hasta que el agente interactúa con ellos.

El ambiente es completamente discreto porque el espacio es un grid y todas las posiciones de los agentes, obstáculos y tiles sucios se especifica como coordenadas (x, y) en el grid.

b) Estructura Física del Ambiente

El ambiente en el que operan los agentes es un espació representado como un grid cuadrangular. El tamaño del grid es customizable, con dimensiones default de 20x20 tiles. Cada tile del grid es una unidad que puede contener uno o más agentes. Y la topología del grid usa la vecindad de Moore, osea que cada tile tiene 8 tiles vecinas. El grid no es toroidal.

1.1 Obstáculos y Estructura

El ambiente contiene obstáculos que delimitan las áreas que se pueden explorar. Hay dos tipos de obstáculos:

- Paredes: En el borde del grid existe una capa de obstáculos que limitan que los agentes salgan del ambiente. Estas paredes se visualizan y definen el espacio.
- Obstáculos Interiores: Dentro del grid se encuentran obstáculos aleatorios. La cantidad de obstáculos se puede customizar como un porcentaje de tiles. Estos obstáculos pueden representar muebles, paredes o interiores.

1.2 Suciedad

Las tiles del ambiente pueden estar en dos estados: limpias o sucias. La suciedad se distribuye de manera aleatoria al inicio de cada simulación en tiles que no contienen obstáculos ni estaciones de carga. El porcentaje de suciedad es customizable. Una vez que un agente limpia una celda, se queda limpia por el resto de la simulación. Cualquier agente que perciba un tile sucio (ya sea su actual o alguna vecina) tendrá acceso a esta información de manera inmediata, pero fuera de ese rango, no sabrá si hay tiles sucias.

1.3 Cargadores

En la simulación 1, existe un único cargador en la posición (1, 1) del grid. En la simulación 2, existen más cargadores (uno por agente), distribuidos aleatoriamente en el grid al inicio de la simulación, con la condición de que cada agente comienza en su propio cargador. Los cargadores son lugares donde los agentes pueden recuperar batería, cuando un agente entra en un tile de cargador (color verde), recuperará batería.

1.4 Batería

Cada agente comienza la simulación con una batería al 100%, la batería se consume y se regenera mediante acciones como:

- Cada movimiento a una celda vecina consume 1%.
- Cada acción de limpiar un tile consume 1%.
- Estar en un cargador incrementan 5%.

La batería termina siendo un recurso que crea un tipo de presión o de aspecto que hay que tomar en cuenta, por ejemplo, un agente con batería baja debe priorizar la búsqueda de una estación de carga, si no, muere.

1.5 Configuración del ambiente

Un aspecto importante del diseño de esta simulación es el ambiente que puede customizarse con parámetros como:

- Dimensiones del grid: width y height.

- Porcentaje de suciedad: dirty_percentage.
- Porcentaje de obstáculos: obstacle_percentage.
- Número de agentes (simulación 2): numAgents.
- Semilla aleatoria: seed

1.6 Simulación 1 y 2

Ambas simulaciones comparten la estructura base del ambiente, pero también tienen diferencias:

- Simulación 1: Un solo agente, un solo cargador, no hay comunicación con otros agentes.
- Simulación 2: Múltiples agentes operando simultáneamente, múltiples cargadores (uno por agente), y un mecanismo de comunicación donde los agentes intercambian información sobre otros cargadores. Además, se implementó un sistema de ocupación que previene que dos roombas estén en el mismo cargador al mismo tiempo.

5. Estadísticas Recolectadas

Para evaluar el desempeño de ambas simulaciones, realicé cinco ejecuciones independientes de cada una utilizando seeds aleatorias, pero manteniendo parámetros idénticos:

- Grid 20x20
- 30% de Suciedad
- 15% de Obstáculos
- Máximo 10000 pasos

En la simulación 1, cuatro de cinco ejecuciones lograron completar la limpieza del 100% de las tiles con tiempos totales variando entre los 1500 y 1600 pasos. El número de movimientos realizados se mantuvo entre los 400 y 480, reflejando diferentes caminos de exploración según la distribución del grid. La batería restante se mantuvo en un rango entre el 35% y el 60%, indicando un consumo y manejo eficiente de batería.
Pero, la quinta ejecución resultó en un fracaso, el agente logró limpiar solo el 31.71% de las tiles antes de quedarse sin batería. Este fallo ocurrió cuando el agente intentaba regresar a su cargador, pero se quedó atrapado en un ciclo de movimientos entre un obstáculo.

En la simulación 2, con dos agentes, todas las cinco ejecuciones alcanzaron el 100% de limpieza, sin errores. Los tiempos fueron más variados, estuvieron entre los 900 y los 2000 pasos, el número total de movimientos entre ambos agentes varió entre 590 y 1500. La batería promedio estuvo entre el 35% y el 70%, indicando una gestión equilibrada de batería entre los dos agentes. La ausencia de fallos en la simulación 2 deja claro que la presencia de más agentes es evidentemente mejor.

Comparando ambas simulaciones, noto patrones en términos de eficiencia. Aunque cada agente realiza aproximadamente 460 movimientos en la simulación 2, el sistema completo termina la limpieza en 1300 en comparación con los 1500 que requiere la simulación 1. Esto representa una mejora en tiempo real de simulación. Un factor que creo que contribuye a que la simulación 2 sea mejor es la comunicación que maneja, cuando dos agentes están cerca,

intercambian las ubicaciones de sus cargadores. Este intercambio permite que ambos agentes desarrollen un conocimiento del ambiente. Las estadísticas recolectadas revelan que la simulación 1 es más eficiente en términos de movimiento por agente, pero la simulación 2 es un sistema superior por su comunicación y el manejo de múltiples agentes que permiten completar la tarea en menos tiempo.

6. Conclusión

Esta actividad fue sin lugar a dudas muy retadora, pero muy interesante de desarrollar. Para empezar, logré demostrar exitosamente el diseño e implementación de sistemas de agentes para cumplir la tarea de limpiar el grid. Los agentes desarrollados fueron capaces de equilibrar objetivos, moverse por el entorno que pueden observar, y en el caso de la simulación 2, coordinarse a través del intercambio de información sin requerir control.

Los resultados experimentales revelaron cosas importantes sobre la eficiencia de los sistemas. La simulación 1 demostró ser más eficiente en términos de movimientos individuales y de gestión de batería, pero mostró una vulnerabilidad de fracaso cuando un agente se atasca por un obstáculo. La simulación 2 alcanzó el 100% en todas las ejecuciones de manera más rápida y consistente. La superioridad de la simulación 2 muestra que múltiples agentes son mejores que 1, nada nuevo; pero también muestra el valor de intercambiar información para coordinar el sistema. Una limitación importante que se me presentó fue la falta de implementar el algoritmo BFS en la simulación 2 debido al tiempo que me quedaba para realizar la actividad. Creo que BFS fue muy efectivo en la simulación 1, permitiendo que el agente encontrara siempre la celda no explorada más cercana. En la simulación 2 dejé la estrategia de selección aleatoria entre vecinos visitados, aunque funcionó bien, creo que haber implementado BFS la segunda parte hubiera resultado en una exploración más eficiente, reduciendo el tiempo por mucho, y mejorando mucho más los tiempos.

Para una versión futura me gustaría implementar un mapeo colaborativo para la segunda actividad, donde los agentes no sólo intercambien las coordenadas de sus cargadores, sino que también compartan mapas de obstáculos. Esto reduciría el trabajo de exploración cuando dos agentes descubren la misma sección.

Este trabajo me permitió entender los sistemas de un solo agente y de muchos agentes coordinados, sus ventajas y sus comportamientos en entornos complejos e impredecibles. Aunque existen muchas áreas de mejora, las soluciones que implementé son sólidas y funcionan de buena forma.