ITESM Campus Santa Fe

Nombre del bloque:

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 301)

Nombre del entregable:

Actividad Roomba

Alumno:

Aquiba Yudah Benarroch Bittán, A01783710

Profesor:

Octavio Navarro Hinojosa

Fecha de entrega:

19 de noviembre de 2025

Informe de Roomba

Problema que se está resolviendo y propuesta de solución

El sistema modela un conjunto de robots tipo Roomba que deben limpiar una habitación representada por MxN espacios donde algunas de estos espacios pueden tener basura y otros pueden ser obstáculos. El objetivo principal de Roomba es limpiar la mayor cantidad de celdas sucias en un tiempo finito, sin quedarse sin batería. Para esto lo que se hizo fue diseñar agentes autómatas que lograran:

- Limpiar celdas sucias
- Navegar evitando obstáculos
- Administrar la batería correctamente
- Regresar a la estación de carga cuando es necesario
- En el caso de la simulación 2, los agentes tipo Roomba se comunican entre sí para compartir información y así reducir el tiempo de limpieza.

Para la simulación 1:

- Un agente Roomba que empieza en la celda [1,1].
- Estación de carga también en [1,1].
- Limpia la basura
- Checa la bateria y con la distancia chebyshev checa si es que necesita empezar a calcular a* para el regreso.
- Evita obstáculos

Para la simulación 2:

- Varios agentes Roomba empiezan en celdas aleatorias.
- Las estaciones de carga se inicializan donde aparecen las Roombas.
- Limpian la basura
- Cuando se encuentran comparten información
- Checan la batería y con la distancia chebyshev checa si es que necesitan empezar a calcular a* para el regreso.
- En caso de que una estación de carga esté ocupada se esperan.

- Evitan obstáculos

En ambos casos Roomba maneja estados.

Diseño de los agentes

Roomba:

- Objetivo: limpiar la mayor cantidad de celdas sucias en un tiempo finito, sin quedarse sin batería.
- Capacidad Efectora:
    - Moverse: Puede desplazarse en las 8 direcciones vecinas.
    - Limpiar: Puede remover basura en la celda actual.
    - Recargar: Recuperar bateria en las estaciones de carga
    - Intercambiar información: Comparte su información con otras Roombas para lograr el objetivo más rápido.
- Proactivo: Es un agente altamente proactivo mediante su máquina de estados:
    1. idle: Estado inicial, evalúa siguiente acción
    2. ready: Preparado para explorar y limpiar
    3. moving: Moviéndose a una celda
    4. cleaning: Limpiando basura
    5. returning: Regresando a la estación de carga
    6. waiting: En caso de que la estación esté ocupada, está esperando.
    7. recharging: Recargando batería
    8. communicating: Intercambiando información con otra Roomba
- Métricas de desempeño
    - Porcentaje de basura a recogida
    - Porcentaje de basura
    - Tiempo límite (steps)
- Percepción
    - La roomba no tiene visión global del ambiente. Su percepción es local, es decir se basa en la celda actual y la de las celdas vecinas donde puede detectar basura, roombas y obstáculos. También conoce la posición de estación de carga y va almacenando las celdas visitadas y aquellas que tienen basura.

- Reactividad:
    - Ajustan sus acciones dependiendo de lo que perciben en el ambiente en cada paso. Si detectan basura en la celda actual, la limpian y si su batería es baja, regresan a su estación.
- Habilidad social:
    - En la simulación de una sola roomba, no hay habilidad social debido a la falta de agentes para interactuar. Sin embargo, en la simulación de varias roombas, si existe habilidad social debido a que pueden comunicarse y pasar información valiosa.
- Racionales:
    - Las roombas son agentes racionales. Actúan con el propósito de limpiar la mayor cantidad de basura en una habitación sin quedarse sin batería. Usan información local, y toman decisiones que intentan maximizar su desempeño. Priorizan acciones en base a la actualidad.

Arquitectura de subsunción de los agentes

Jerarquía de comportamiento (de mayor a menor prioridad)

1. Sobrevivir
2. Recargarse
3. Limpiar basura
4. Moverse a celdas no visitadas
5. Moverse a celdas que conoció en el regreso que tienen basura.

Características del ambiente

- Parcialmente accesible: Las Roombas sólo pueden ver la información de su celda y sus vecinos, sin embargo no de lo que está pasando más lejos de esto.
- No determinista: Las acciones como la cantidad de basura y obstáculos dependen de la aleatoriedad y eso no asegura que siempre se tenga el mismo resultado.

- No episódico: Los agentes toman sus decisiones a partir de episodios actuales, es decir, el paso actual.
- Estático: Los obstáculos y basura no van creciendo o cambiando, sino que más bien una vez que empieza la simulación se quedan igual hasta ser limpiados.
- Discreto: Hay un número finito de opciones que se pueden hacer, es decir hay un número finito de estados para la Roomba.

PEAS

- Performance: La roomba es capaz de limpiar basura, explorar nuevas celdas y recargarse.
- Environment: Ambiente compuesto por un grid de tamaño M x N celdas, donde cada una de estas puede estar vacía, tener basura, obstáculos, estación de recarga.
- Actuators: Las roombas son capaces de moverse a celdas, limpiar basura, evitar obstáculos y recargarse.
- Sensors: Puede percibir información de su celda actual y sus vecinos para ver si hay basura, obstáculos o otras roombas.

Estadísticas de las simulaciones

Simulación 1

Para cada simulación se imprime en la consola 4 estadísticas:

- Porcentaje de basura que se limpió.
- Batería con la que acaba Roomba la simulación
- La cantidad de pasos del Roomba
- El tiempo de simulación (steps)

Con estas 4 estadísticas se puede observar la eficiencia del Roomba de limpiar el cuarto y ver realmente cual es el comportamiento de esta. Además se trataron diferentes parámetros para cada simulación y poder observar cómo Roomba hacía la simulación en esas condiciones.

Simulación 1A

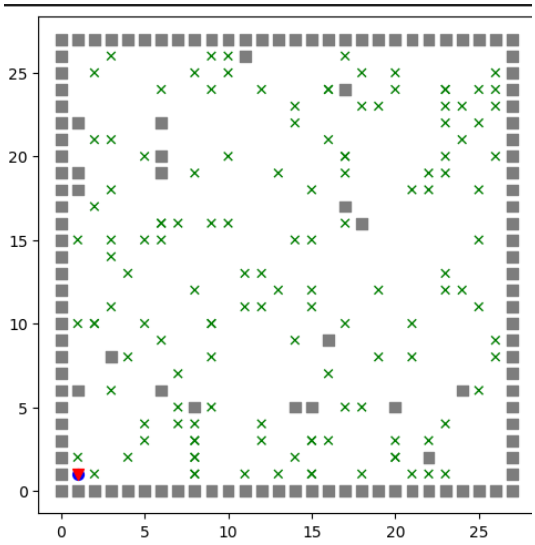
Parámetros utilizados:

- Número de obstáculos: 20

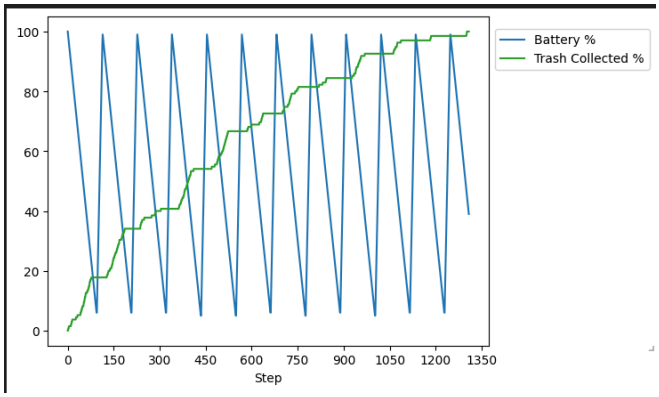
- Basura: 0.2%

- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
Trash Collected %: 100.00
Battery %: 39.00
Roomba Steps: 964.00
Total Time (steps): 1308.00
```

Análisis de resultados:

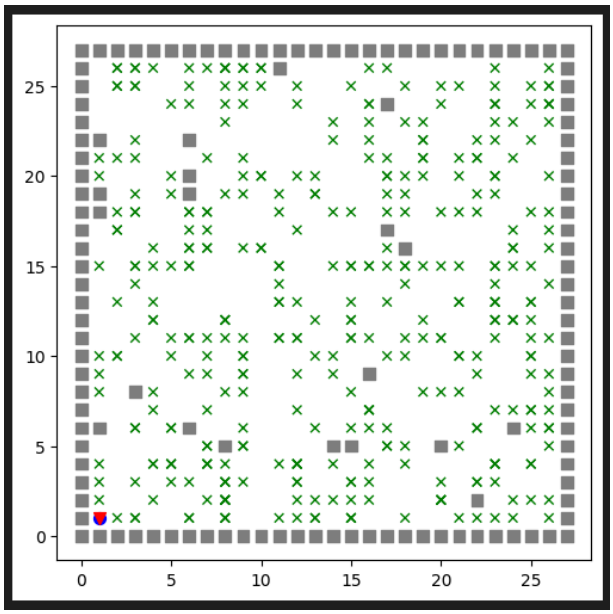
En esta simulación, la Roomba muestra que es eficiente debido a que avanza limpiando por el mapa mientras administra su energía sin riesgos. Esto lo logra debido a que alterna correctamente entre exploración y recarga. La gráfica muestra cómo Roomba tiene sus ciclos de carga, lo que representa que sabe cuándo es momento de regresar y cuándo puede seguir buscando basura. Finalmente alcanza el 100% de la basura recogida antes del límite de pasos. demostrando que incluso con una cantidad moderada de obstáculos y poco porcentaje de basura puede limpiar todo el cuarto de forma eficiente.

Simulación 1B

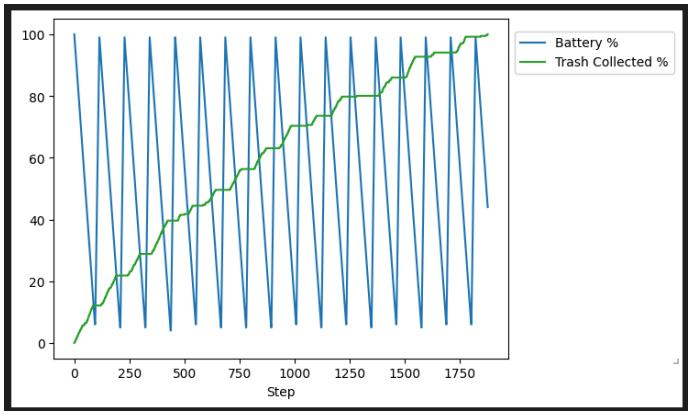
Parámetros utilizados:

- Número de obstáculos: 20
- Basura: 0.55%
- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
Trash Collected %: 100.00
Battery %: 44.00
Roomba Steps: 1199.00
Total Time (steps): 1875.00
```

Análisis de resultados:

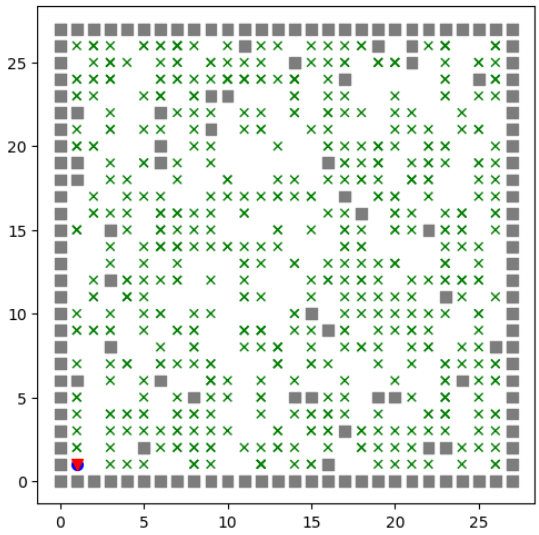
En esta simulación, Roomba mantiene el mismo comportamiento eficiente al limpiar el cuarto con basura. Avanza de manera constante mientras administra su batería. Conforme avanza, el porcentaje de basura recogida también aumenta hasta lograr completar la tarea y haber recogido un 100% de la basura. Esto es una demostración de que aunque exista mayor basura, Roomba es capaz de mantener el ritmo de limpieza continuo y eficiente.

Simulación 1C

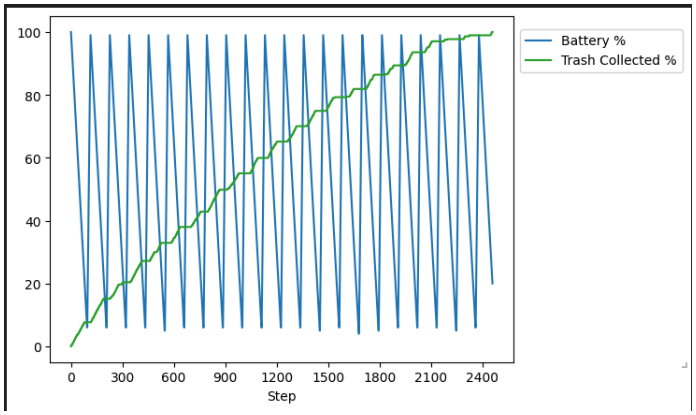
Parámetros utilizados:

- Número de obstáculos: 40
- Basura: 0.85%
- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
Trash Collected %: 100.00
Battery %: 20.00
Roomba Steps: 1486.00
Total Time (steps): 2460.00
```

Análisis de resultados:

En esta última simulación, Roomba enfrenta un cuarto con el doble de obstáculos que en las simulaciones anteriores y mucho mayor porcentaje de basura. Aquí se puede observar en la gráfica que Roomba requiere de mayor cantidad de recargas debido al entorno al que se encuentra. No obstante, logra limpiar el cuarto dentro del tiempo límite establecido. Podemos ver, que en términos de pasos de Roomba, no hubo tanta diferencia con las otras simulaciones. Sin embargo, cuando hablamos de tiempo podemos ver la diferencia substancial, que justamente se genera debido a la frecuencia de recargas para esta última simulación.

Simulación 2

Para cada simulación se imprime en la consola 5 estadísticas:

- Número de Roombas vivas
- Porcentaje de basura recolectada
- El tiempo de simulación (steps)
- La batería promedio de las Roombas
- La cantidad de pasos de las Roombas promedio

Con estas 5 estadísticas se puede observar la eficiencia de los Roombas al limpiar el cuarto y ver realmente cual es el comportamiento de estas. Además se trataron diferentes parámetros para cada simulación y poder observar cómo el hecho de tener varios agentes, afectaba la simulación.

Simulación 2A

Parámetros utilizados:

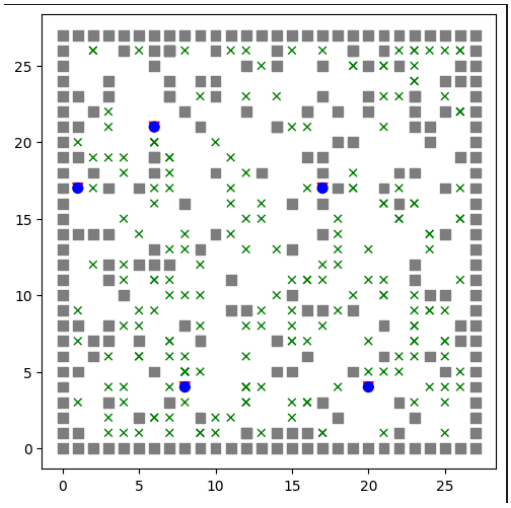
- Número de Roombas: 5

- Obstáculos: 0.2%

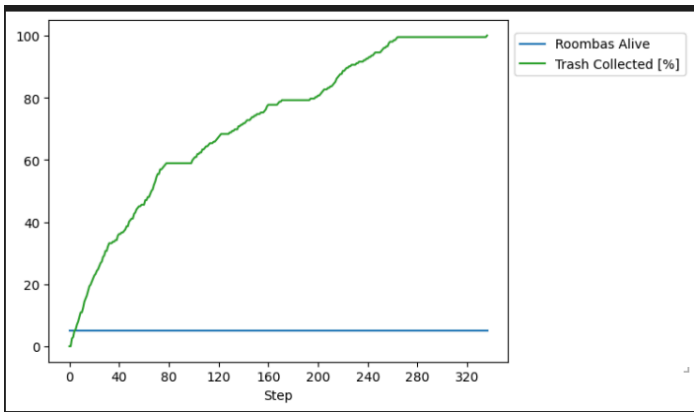
- Basura: 0.3%

- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
     Roombas Alive  Trash Collected [%]  Time (Steps)  Battery %  Roomba Steps
336              5                100.0           336       64.0         231.8
Individual Roomba Stats:
  Roomba 110: Battery 65%, Steps 233
  Roomba 112: Battery 50%, Steps 248
  Roomba 114: Battery 84%, Steps 229
  Roomba 116: Battery 50%, Steps 245
  Roomba 118: Battery 71%, Steps 204
```

Análisis de resultados:

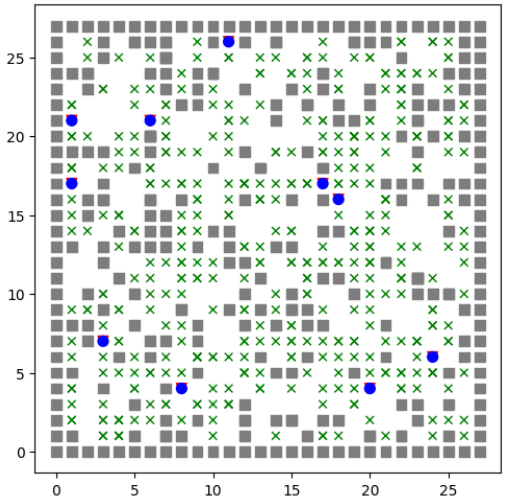
En esta simulación, tenemos a 5 Roombas que se enfrentan a un cuarto con baja cantidad de basura y obstáculos. En la gráfica se puede observar como ninguna de las Roombas se murió y todas cumplieron el objetivo de limpiar el cuarto. Debido a que tenemos 5 agentes, podemos observar como a comparación de la simulación 1, el tiempo para limpiar el cuarto bajó considerablemente. Además, debido a la gráfica podemos saber que la función de que cuando una estación de carga está ocupada y la Roomba se queda esperando a que se desocupe, funciona correctamente.

Simulación 2B

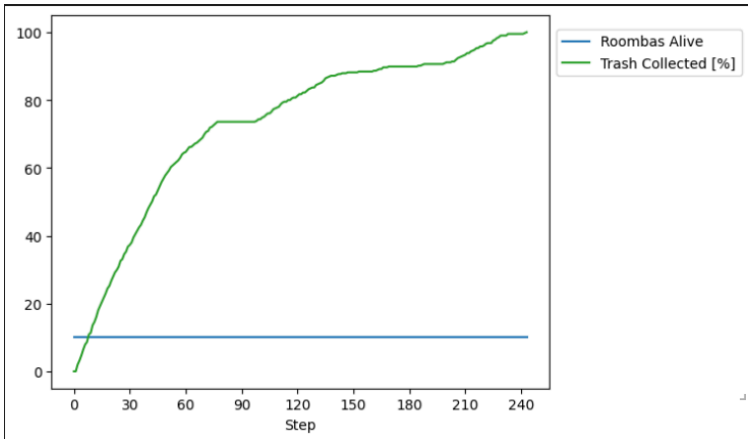
Parámetros utilizados:

- Número de Roombas: 10
- Obstáculos: 0.3%
- Basura: 0.6%
- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
     Roombas Alive  Trash Collected [%]  Time (Steps)  Battery %  Roomba Steps
243             10                100.0           243       64.4         146.0
Individual Roomba Stats:
  Roomba 110: Battery 64%, Steps 134
  Roomba 112: Battery 62%, Steps 162
  Roomba 114: Battery 47%, Steps 133
  Roomba 116: Battery 97%, Steps 132
  Roomba 118: Battery 48%, Steps 126
  Roomba 120: Battery 78%, Steps 139
  Roomba 122: Battery 47%, Steps 158
  Roomba 124: Battery 80%, Steps 136
  Roomba 126: Battery 74%, Steps 169
  Roomba 128: Battery 47%, Steps 171
```

Análisis de resultados:

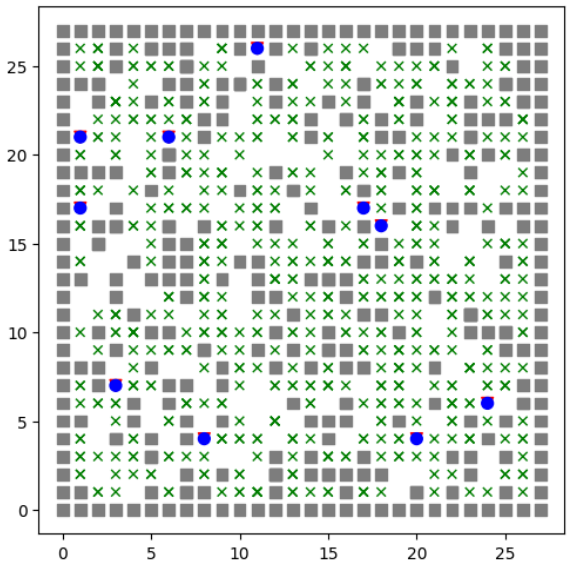
En esta simulación, las Roombas mantienen el mismo comportamiento eficiente al limpiar el cuarto con basura. En términos de tiempo, debido a que la cantidad de Roombas aquí es el doble, tardan mucho menos para limpiar la habitación completa. Todavía incluso con estas condiciones, las Roombas logran recargarse sin morir.

Simulación 2C

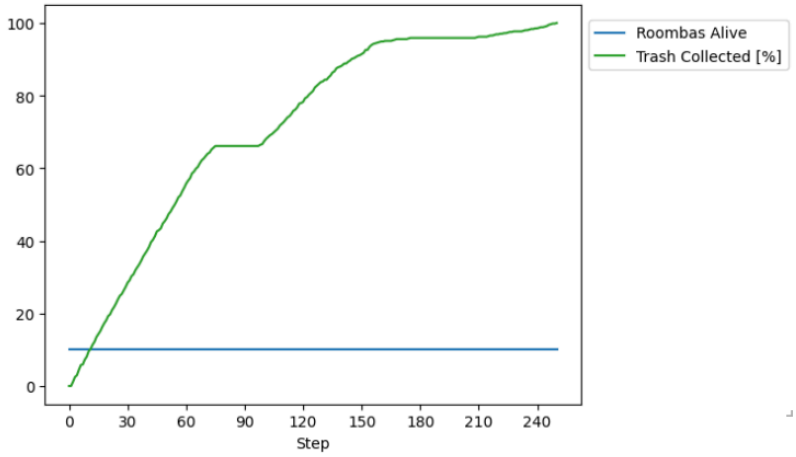
Parámetros utilizados:

- Número de Roombas: 10
- Obstáculos: 0.35%
- Basura: 0.9%
- Límite de pasos: 3000

Visualización del grid:



Resultados:




```
SIMULATION FINAL STATS
     Roombas Alive  Trash Collected [%]  Time (Steps)  Battery %  Roomba Steps
250             10                100.0           250       50.9         138.6
Individual Roomba Stats:
  Roomba 110: Battery 40%, Steps 132
  Roomba 112: Battery 47%, Steps 158
  Roomba 114: Battery 40%, Steps 138
  Roomba 116: Battery 42%, Steps 149
  Roomba 118: Battery 56%, Steps 158
  Roomba 120: Battery 55%, Steps 120
  Roomba 122: Battery 56%, Steps 131
  Roomba 124: Battery 60%, Steps 130
  Roomba 126: Battery 72%, Steps 119
  Roomba 128: Battery 41%, Steps 151
```

Análisis de resultados:

En esta última simulación, las Roombas enfrentan un cuarto con muchos obstáculos y el máximo porcentaje de basura posible. Aun así, siguen siendo muy eficientes en términos de tiempo y limpieza, logrando limpiar el cuarto en 250 steps. Podemos también observar que la limpieza de la basura crece rápidamente hasta llegar a un punto medio donde se estabiliza un poco debido a la recarga de las roombas, para después volver a crecer. Al final, se estabiliza de nuevo hasta poder limpiar el cuarto completo y completar la tarea.

Conclusiones

A partir de las gráficas obtenidas en ambas simulaciones, es posible identificar fortalezas del modelo y áreas de mejora en la administración de batería y exploración. En primer lugar, observando la gráfica de la Simulación 1 se puede apreciar que el consumo de energía es un patrón. Se puede observar que la Roomba se descarga pero siempre logra regresar a su estación de carga. En otras palabras, mantiene siempre su comportamiento de supervivencia, regresando a la estación antes de morir. Con respecto a la basura, se puede observar que el porcentaje va creciendo rápidamente al principio debido a que encuentra la basura cerca de ella. No obstante, una vez que tiene que ir a celdas más lejanas, el porcentaje de basura recogida se estabiliza.

Con respecto a la segunda simulación, se puede observar como el hecho de tener más de una Roomba ayuda a que el tiempo de limpieza se reduzca significativamente. Además con la función de esperar en el caso de que una estación esté ocupada evitamos que las Roombas se mueran. Aquí también el crecimiento de la basura recogida tiene un comportamiento esperado: al inicio sube rápidamente, pero conforme pasa el tiempo los incrementos se vuelven más pequeños y lentos.

Lo que se hizo al inicio para la simulación 1 es que en cada step calcule A* para obtener el mejor camino de regreso a la estación de carga. Aunque esto funcionó correctamente para la simulación 1, después de analizarlo con el profesor, nos dimos cuenta que no iba a ser tan efectivo cuando se tengan muchos agentes actuando en la misma simulación. Esto debido a que realmente no era necesario calcular A* en cada paso, sino que más bien debíamos hacerlo cuando se cumpla cierto porcentaje de batería restante. Después de investigación, encontramos

que la distancia de Chebyshev era la indicada. Lo que se hizo fue que checamos nuestra batería en cada paso y calculamos la distancia con Chebyshev para poder definir cuándo usar A*. Esto optimizó el código y la carga de procesamiento para este mismo.

Una de las razones por las que en la simulación 2 el tiempo de ejecución es muy corto es debido a la habilidad social de compartir información entre roombas. Gracias al cambio de información de celdas visitadas y estaciones de carga logramos optimizar el tiempo en que las Roombas pueden limpiar el cuarto.

Una de las mejoras que se podría hacer para optimizar mucho más la limpieza tanto en la primera como en la segunda simulación es que Roomba al igual que guarda las celdas vecinas que son basura, que también tenga la posibilidad de guardar las celdas vecinas que no son nada. Esto para que en futuros pasos sepa que no va a haber nada en esas celdas por lo que explora otras. Es importante mencionar que el hecho de que los vecinos no hayan sido nada, no significa que no pase por ahí. Sin embargo, se puede usar al principio de la simulación para que Roomba tenga más batería y explore más celdas nuevas, optimizando así el tiempo de limpieza.

Otra mejora que se puede hacer es que cuando existan varios agentes se utilicen técnicas como divide y conquista para completar la limpieza más rápido. La propuesta sería dividir el tablero por secciones dependiendo del número de Roombas y su posición. Esto para que cada una de ellas limpie su sección asignada y así evitar que alguna esté esperando por otra. Además, en caso de que alguna acabe con su sección, entonces que esta pase a la sección más cercana para buscar basura y ayudar a limpiar la habitación. Aquí es justamente donde entran las funcionalidades actuales de comunicación y esperar a una estación de carga x.

En resumen, las simulaciones muestran que el sistema es funcional, pero aún tiene espacio para muchas mejoras. La arquitectura actual cumple las funcionalidades que se piden y garantiza que el cuarto se limpie dentro de un tiempo finito. Para futuras versiones nos podríamos enfocar en maximizar la batería con técnicas para también minimizar el tiempo de limpieza.