ITESM Campus Santa Fe

Nombre del bloque:

TC2008B: Modelación de sistemas multiagentes con gráficas computacionales (Gpo 302)

Nombre del entregable:

Informe Actividad Roomba

Alumno:

Julio César Rodríguez Figueroa, A01029680

Profesores:

Octavio Navarro Hinojosa

Fecha de entrega:

18 de noviembre de 2025

Problema que se está resolviendo, y la propuesta de solución.

En esta simulación se nos presenta una sala de MxN casillas las cuales estarán distribuidas en 4 tipos, casillas vacías, casillas sucias, casillas de obstáculos y estaciones de recarga. Esto con el fin de pueda ser limpiado dentro de un límite de tiempo dado por agentes que simulan ser Robots Roomba maximizando el número de celdas limpias antes de que se acabe el mismo. La propuesta para solucionar este problema será la siguiente, implementar una simulación basada en agentes de limpieza de la clase de Robots rumba que serán capaces de poder investigar el entorno, cargarse, limpiar basura. Se compararon 2 escenarios en esta simulación. Un solo robot con una sola estación de carga y un conjunto de los mismos que podrán interactuar.

El diseño de los agentes

Objetivo

Cada uno de los agentes en las simulaciones tendrá como objetivo principal el limpiar el mayor número de celdas antes de que termine el tiempo máximo de ejecución. Si es que el o los agentes se quedan sin batería deberá detenerse la simulación.

Capacidad Efectora

Cada uno de los mismos podrá moverse a cualquier celda que no sea un obstáculo en las 8 direcciones que posee (arriba, abajo, izquierda, derecha, arriba derecha, arriba izquierda, abajo, izquierda, abajo derecha).

Limpiar la celda sucia en donde esté.

Determinar la decisión para regresar a una estación de recarga.

Quedarse en la estación de recarga para recuperar energía.

En el caso de los múltiples agentes, estos podrán comunicarse para poder descubrir ubicaciones de las diferentes estaciones de carga.

Percepción

Cada uno de los agentes podrá revisar los vecinos para poder determinar hacia dónde moverse, a su vez podrá percibir su nivel de batería actual.

En el caso del escenario con múltiples agentes se deberá poder percibir las demás estaciones de recarga.

Proactividad

El agente actúa de manera autónoma al tomar las decisiones de la limpieza, de desplazamiento y de recarga sin instrucciones externas.

Se deben de establecer umbrales en la batería de los agentes y en un cálculo constante de la distancia hacia una estación de recarga.

Métricas de desempeño

Se recopilará la siguiente información del modelo:

- Cuando terminó la simulación en pasos, ya sea si se limpio todo el grid o el
- Número de movimientos realizados
- Número de celdas limpiadas
- Número de celdas sucias pendientes
- Batería restante del agente

En el caso de la simulación de muchos agentes se hará un promedio de la batería restante de los mismos.

La arquitectura de subsunción de los agentes.

Se usará una jerarquía para poder organizar de mejor manera el comportamiento de los agentes.

1. Supervivencia del agente (Prioridad máxima):
    a. Si la batería es baja (aproximadamente 20%): Interrumpe cualquier acción y navega hacia la estación de carga más cercana.
    b. Si esta sobre la estación: Recuperar suficiente energía permaneciendo quieto
2. Limpieza:

    a. Si la celda en la que esta el agente esta sucia: Limpiar inmediatamente y gastar 1% de bateria
3. Exploración:
    a. En caso de que no haya celdas sucias cerca: Seleccionar una celda aleatoria vecina a la cual moverse que no haya sido explorada.
    b. Evitar obstáculos.
4. Colaboración (Prioridad baja y solo usada en la simulación con múltiples agentes):
    a. Permitir que los agentes puedan compartir las ubicaciones de las estaciones de carga existentes.
    b. Evitar colisiones moviéndose a celdas que no estén ocupadas por algún otro agente.

Características del ambiente.

El ambiente estará conformado en nuestra sala de MxN por los 3 tipos de celda mencionados en la descripción del problema, casillas vacías o limpias, casillas de obstáculo y estación de recarga. Adicional a esto contará con variables de control como el tiempo máximo de ejecución. Siendo un ambiente parcialmente accesible, ya que a pesar de que los agentes empiezan solo con el conocimiento de su celda al explorar se hará más consciente de su ambiente. Siendo un ambiente estático ya que no cambia solo y cambia en base a las acciones del ambiente.

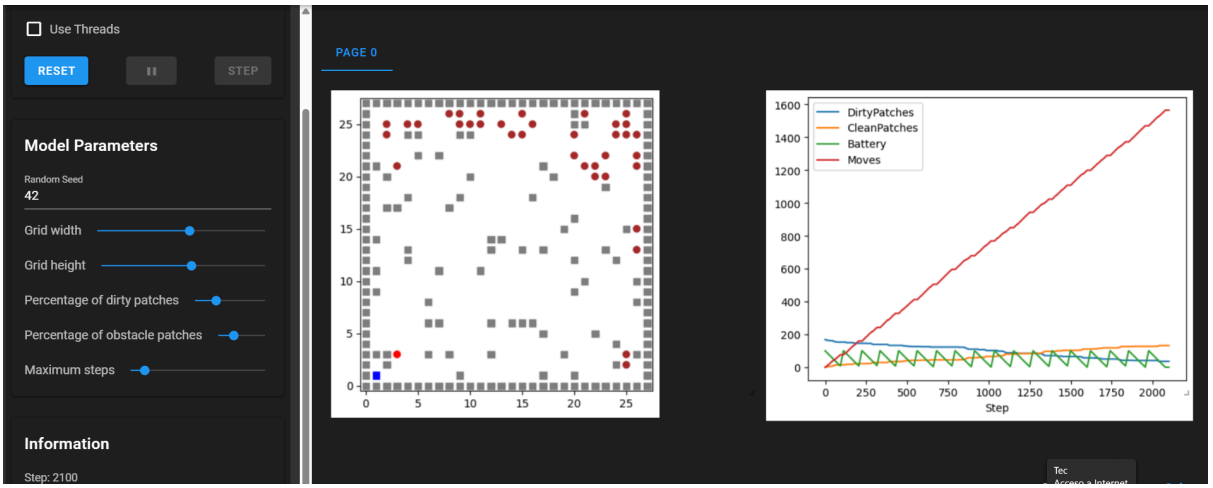
Otra consideración de nuestro ambiente es que en este caso, M y N tendrán el mismo valor para hacer un grid cuadrado donde el agente se podrá mover en un área de M-1 y N-1 dependiendo de los obstáculos aleatorios.

Las estadísticas recolectadas en las simulaciones.

Para la primera simulación de un solo agente se empezaron con los siguientes parámetros:

- Batería inicial: 100
- Nivel de batería que se considera crítico: 20
- Porcentaje de obstáculos: 0.05 de un máximo de 3
- Porcentaje de celdas sucias: 0.3 de un máximo de 1
- Número máximo de pasos de la simulación 3000

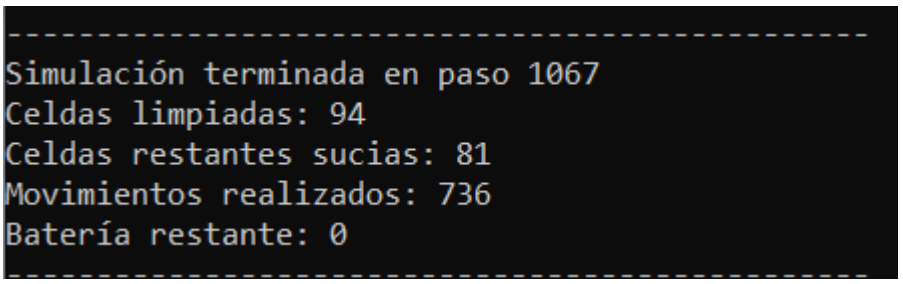
Se obtuvieron los siguientes resultados al acabar la simulación:




```
----------------------------------------------------
Simulación terminada en paso 2100
Celdas limpiadas: 132
Celdas restantes sucias: 36
Movimientos realizados: 1565
Batería restante: 0
----------------------------------------------------
```

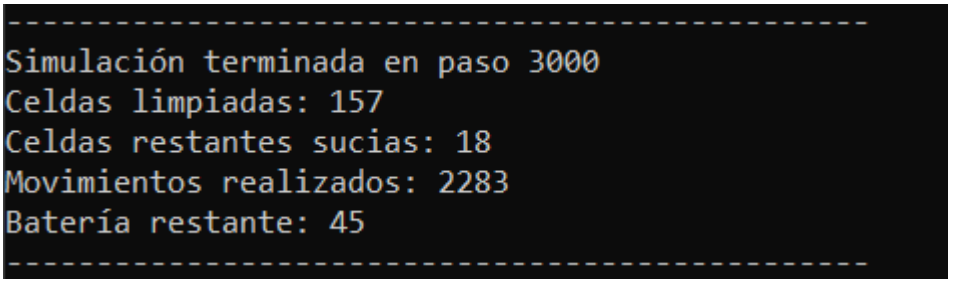
En base a estos resultados se tomó en cuenta la posibilidad de poder cambiar el porcentaje para identificar si la batería esta en un nivel crítico será cambiado a 30, dejando el resto de parámetros igual.

Siendo estos los resultados obtenidos.



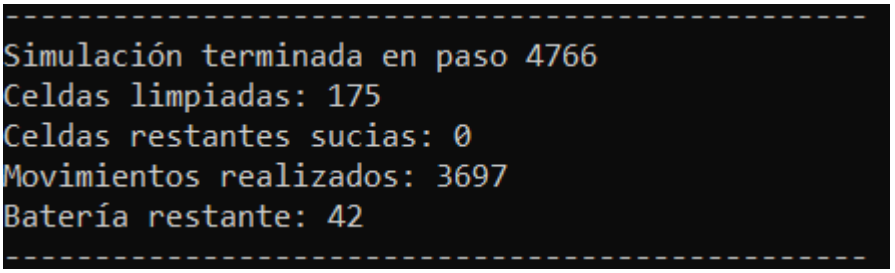
Por lo que se tomó la decisión de implementar un cálculo de distancia dinámico a su base de recarga con base al algoritmo de A* con cada paso que de y en caso de pasar el umbral de la batería regresará a recargarse, de igual forma se estableció un margen de batería para llegar al destino y evitar que llegue muerto el agente, el margen fue de 3.

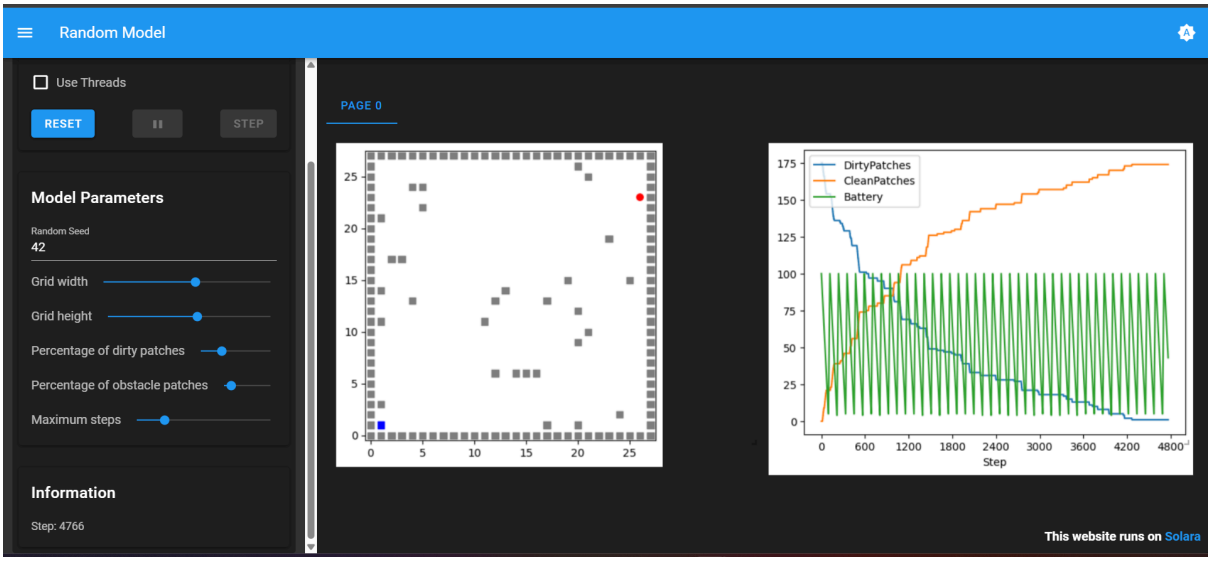
Estos fueron los resultados:



Por lo que se debe de ajustar el número de pasos si es que se quiere que se limpie de manera completa el cuarto por un solo roomba. Por ejemplo 5000, aquí se presentan los resultados del modelo con los parámetros:

- Batería inicial: 100
- Nivel de batería que se considera crítico: 30
- Porcentaje de obstáculos: 0.05 de un máximo de 3
- Porcentaje de celdas sucias: 0.3 de un máximo de 1
- Número máximo de pasos de la simulación 5000

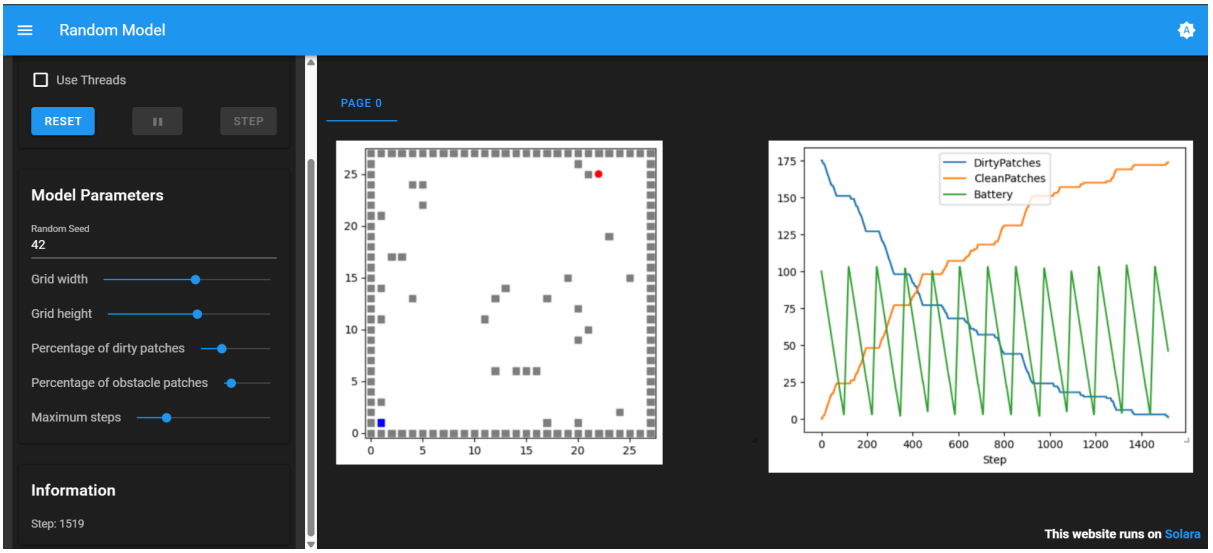




Para poder optimizar este proceso se debe mejorar el movimiento del robot para que explore de manera más eficiente. Actualmente, el robot roomba posee un movimiento mayormente aleatorio, explorando los vecinos y priorizando los que están sucios, sin embargo en el momento que ya no haya vecinos sucios y sin vistar empieza a moverse aleatoriamente. Para lo que se propone el siguiente comportamiento cuando esté en el estado “Explorando”:

En el método de move() se implementan todas las acciones en esta jerarquía, primero revisar los vecinos que tiene en ese momento, en prioridad el buscar celdas sucias, en caso de que si, moverse a una aleatoria. En caso de que no haya celdas sucias, busca en las que no haya visitado para poder moverse cerca a esas celdas. Después se podrá mover si tiene alguna celda pendiente para poder seguir explorando, y por último si no entra en alguno de esos casos se moverá aleatoriamente como en la simulación anterior.

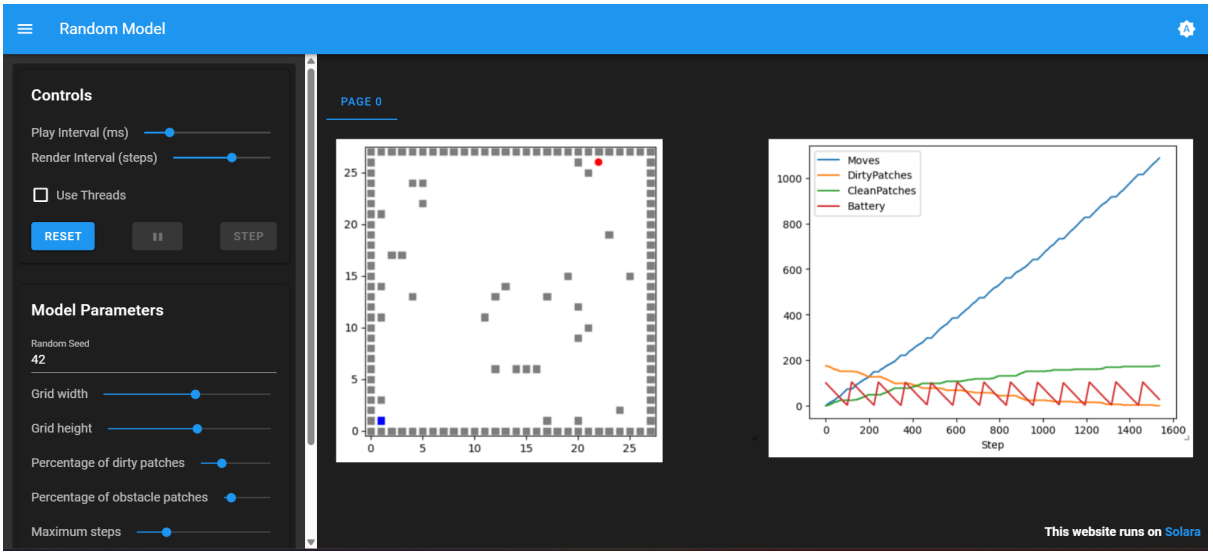
Lo que nos dio los siguientes resultados en la simulación con los mismos parámetros de la anterior:




```
------------------------------------------------
Simulación terminada en paso 1519
Celdas limpiadas: 175
Celdas restantes sucias: 0
Movimientos realizados: 1070
Batería restante: 45
------------------------------------------------
```

Lo que representa una mejora considerable con respecto al tiempo de limpieza, lo que indica que el tiempo máximo que se podría establecer para completar la simulación son 1600 o 2000 pasos de simulación.

Se realizó una simulación más para poder ver los movimientos realizados, conservando los mismos parámetros:




```
----------------------------------------------------
Simulación terminada en paso 1519
Celdas limpiadas: 175
Celdas restantes sucias: 0
Movimientos realizados: 1070
Batería restante: 45
----------------------------------------------------
```

Parte 2

Para la simulación con múltiples robots, los parámetros iniciales serán los siguientes:

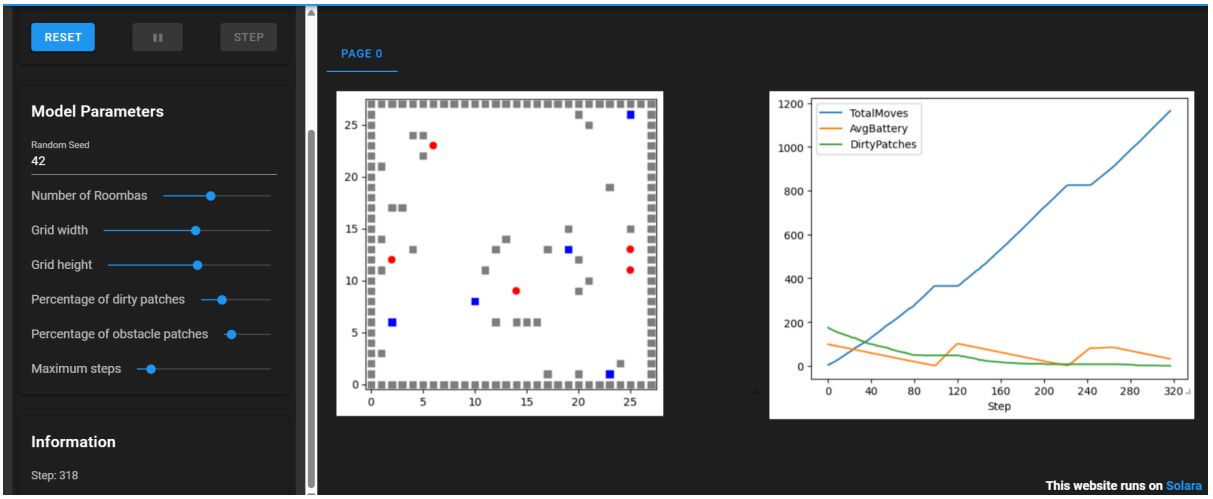
- Número de agentes: 5
- Batería inicial: 100
- Nivel de batería que se considera crítico: 30
- Porcentaje de obstáculos: 0.05 de un máximo de 3
- Porcentaje de celdas sucias: 0.3 de un máximo de 1

Usando el mismo comportamiento de la simulación de un agente, con la diferencia de que ahora pueden compartir sus conocimientos. Además de que la simulación acabará cuando mueran todos los roombas. Aquí los resultados de la simulación:

```
------------------------------------------------
Simulación terminada en paso 528
Celdas limpiadas: 175
Celdas restantes sucias: 0
Movimientos realizados: 2005
Batería restante promedio: 64.0
------------------------------------------------
```

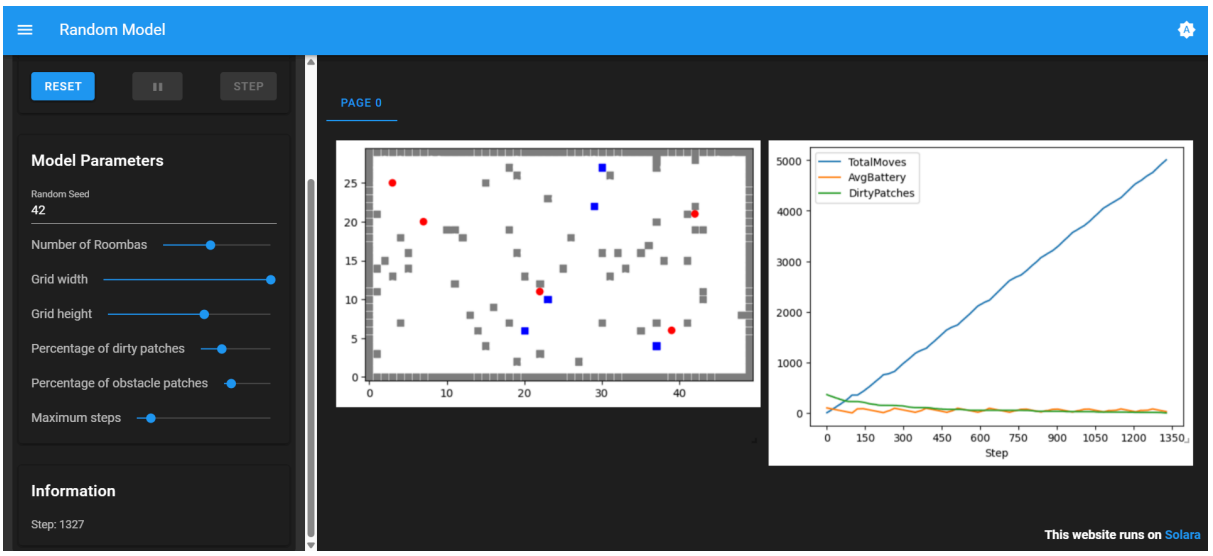
Tras finalizar la simulación pude notar comportamientos extraños a la hora de hacer que compartan conocimiento, siendo que había agentes que se teletransportan, además de esto, varios agentes podrán recargarse en la misma estación, lo cual no debe pasar. Para solucionar el movimiento se cambió un poco el método move(). En el caso de los agentes para ir a otras estaciones de carga, ahora si es que no conocen ninguna estación de carga y la suya esta ocupada se quedará esperando su turno para cargar.

Con estos cambios se obtuvieron los siguientes resultados:




```
------------------------------------------------
Simulación terminada en paso 318
Celdas limpiadas: 175
Celdas restantes sucias: 0
Movimientos realizados: 1164
Batería restante promedio: 32.2
------------------------------------------------
```

Posteriormente se hizo una prueba con un grid más grande (50x30) para poder ver cuánto se tardaba la misma cantidad de robots. Estos fueron los resultados:




```
--------------------------------------------------
Simulación terminada en paso 1327
Celdas limpiadas: 363
Celdas restantes sucias: 0
Movimientos realizados: 5005
Batería restante promedio: 26.4
--------------------------------------------------
```

Conclusiones

Tras revisar las simulaciones, he podido notar que gracias a las optimizaciones realizadas en el método de “move()” es que se puede optimizar el tiempo de limpieza priorizando las casillas que encuentre que estén sucias y después esos lugares que quedaron pendientes. Además de que el uso de A* para poder volver a recargarse es eficiente, sin embargo al estar basado únicamente en el grafo que posee el robot en su memoria pues afecta en la eficiencia del algoritmo, entre menos haya explorado más vueltas dará y tardará un poco más de movimientos en llegar.

También se pudo ver que el aumentar el número de agentes ayuda a la eficiencia, sin embargo, los agentes que se quedan esperando afectan ligeramente a la eficiencia de la limpieza. Otro punto de la simulación de multiagentes es que el compartir conocimiento entre agentes permitió el poder explorar de forma más eficiente, ayudando a la eficiencia de la misma.

Por último las pruebas con grids más grandes probaron que el comportamiento programado en nuestra máquina de estados posee un rendimiento aceptable. Lo que prueba que se puede utilizar este comportamiento en ambientes más grandes.

Como trabajo futuro se podría implementar una manera de recorrer el grid por filas o columnas para poder tener una exploración más fluida además de un mejor trabajo de limpieza cuando quede menos suciedad. En el caso de tener muchos agentes, sería delimitar dependiendo de su posición inicial y del número de los mismos delimitar áreas para poder limpiar de forma más ordenada, y en caso de terminar antes un área se debe poder explorar libremente en los límites de las mismas usando los métodos implementados en estas simulaciones.