Instituto Tecnológico y de Estudios Superiores de Monterrey.
Campus Estado de México

Modelación de sistemas multiagentes con gráficas computacionales
TC2008B | (Gpo 301)

M1 Actividad

Informe de Simulación del Modelo de Limpieza

Profesores a cargo

Jorge Adolfo Ramírez Uresti

Manuel Olmos Antillón…..…… ITC | A01750748

Yael Octavio Pérez Méndez…..……... …… ITC | A01799842

Fecha de entrega: 8 de Noviembre del 2024

Introducción

El objetivo de este informe es analizar la efectividad de un modelo de limpieza automatizado en un entorno de MxN celdas. El modelo utiliza agentes que se desplazan por la habitación limpiando celdas sucias. Los parámetros principales de la simulación incluyen el tamaño de la habitación, el número de agentes, el porcentaje de celdas inicialmente sucias y el tiempo máximo de ejecución. El análisis incluirá la recopilación de datos relevantes durante la ejecución, como el tiempo necesario para limpiar todas las celdas, el porcentaje de celdas limpias al finalizar la simulación y el número de movimientos realizados por los agentes.

Desarrollo

La simulación sigue los siguientes pasos:

1. Inicialización
   - Se crean celdas sucias en ubicaciones aleatorias dentro del grid.
   - Todos los agentes comienzan en la celda (1,1).

2. Ejecución de la Simulación
   - En cada paso de tiempo, los agentes verifican si su celda actual está sucia. Si es así, la limpian.
   - Si la celda está limpia, el agente se mueve a una de las 8 celdas vecinas de manera aleatoria.

3. Recopilación de Datos
   - Tiempo necesario para limpiar todas las celdas o alcanzar el tiempo máximo.
   - Porcentaje de celdas limpias al final de la simulación.
   - Número de movimientos realizados por todos los agentes.

El código implementado para esta simulación está estructurado en una clase `CleaningModel` que maneja la inicialización, movimiento y limpieza de las celdas, controlado a través de funciones de visualización y control de tiempo.

Resultados y Observaciones

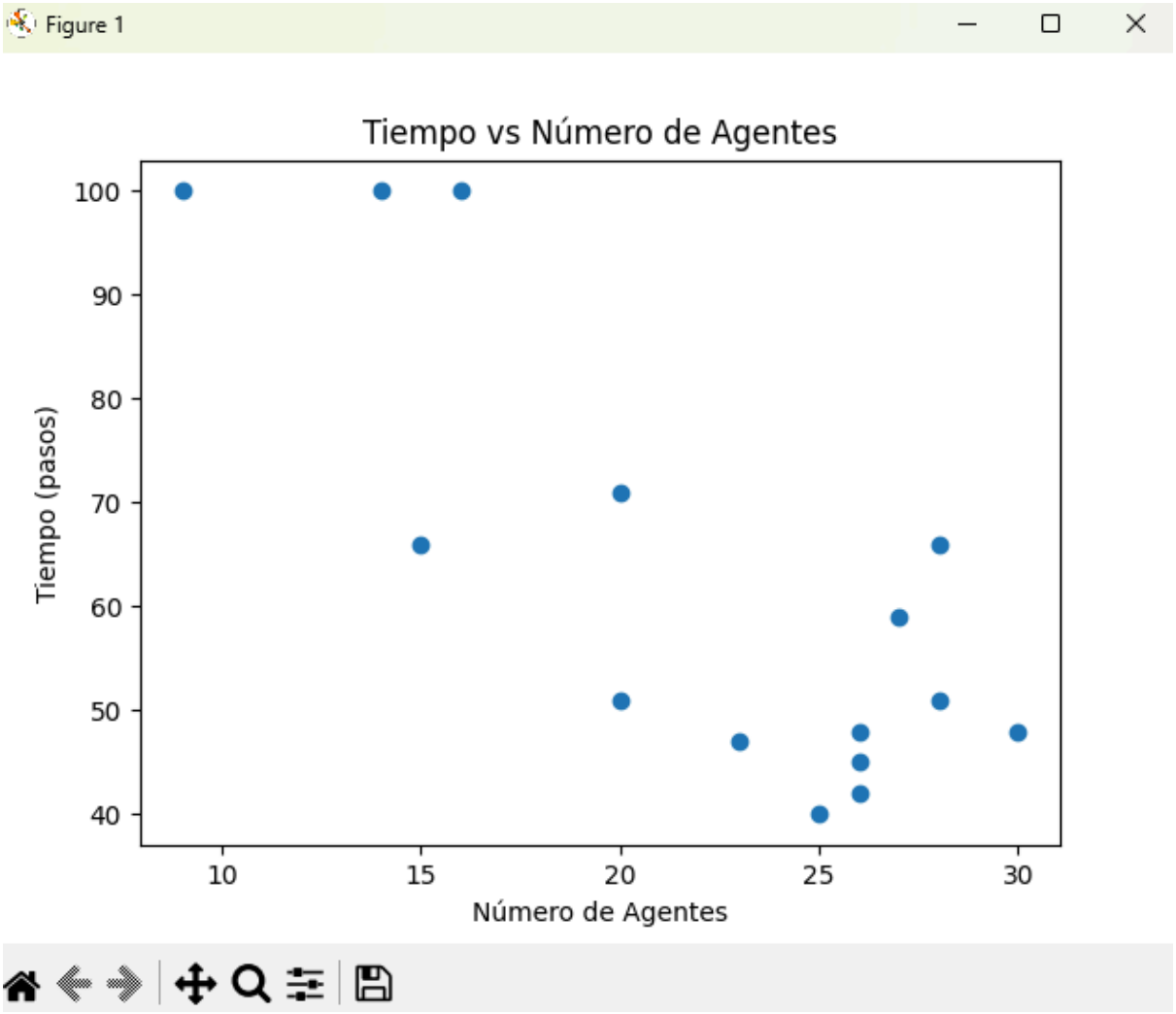
Para un ejemplo con 10 agentes, 30% de celdas sucias y un máximo de 100 pasos, los resultados observados fueron:

- Tiempo Necesario para Limpiar: El tiempo varía dependiendo de la disposición aleatoria de las celdas sucias y los movimientos de los agentes.
- Porcentaje de Celdas Limpias: Esto depende del porcentaje inicial de celdas sucias y la eficiencia del movimiento de los agentes.
- Número de Movimientos: Se correlaciona directamente con el número de agentes y el número de pasos tomados.

Al analizar el impacto del número de agentes, se observó que:

- Aumentar el número de agentes generalmente disminuye el tiempo necesario para limpiar todas las celdas debido a los esfuerzos paralelos de limpieza.
- Sin embargo, más agentes también incrementan el número de movimientos, lo que podría resultar en redundancia donde múltiples agentes limpian las mismas celdas.

Ahora realizando 15 iteraciones con diferente número de agentes aleatoriamente inicializados obtenemos que:



La anterior gráfica nos muestra una tendencia y concentración en la parte inferior izquierda de la gráfica, lo que nos dice que a mayor número de agentes menor número de pasos de tiempo se utilizarán para limpiar la celdas.

Conclusión

1. Impacto del Número de Agentes:
   - Incrementar el número de agentes tiende a reducir el tiempo necesario para la limpieza completa, pero también puede incrementar movimientos redundantes.
2. Eficiencia:
   - La eficiencia de la limpieza mejora con un número equilibrado de agentes que pueden cubrir diferentes áreas sin solaparse.
   - Es recomendable determinar un número óptimo de agentes basado en el tamaño del grid y el porcentaje de celdas sucias para maximizar la eficiencia de limpieza.

Recomendaciones

- Para Grids Grandes: Usar un mayor número de agentes para asegurar una limpieza más rápida.
- Para Grids Pequeños: Menos agentes pueden ser más eficientes para evitar redundancias.
- Simulaciones Futuras: Realizar múltiples ejecuciones con diferentes parámetros para analizar estadísticamente el rendimiento y determinar configuraciones óptimas.

Trabajo Futuro

- Implementar algoritmos de movimiento más sofisticados para minimizar movimientos redundantes.
- Explorar estrategias de colaboración entre agentes para mejorar la eficiencia.

Para más detalles, puede referirse al [código fuente]
https://github.com/Yael-PM/M1_Actividad/blob/main/Clean_Agent.py