Trabajo Final (TF): Tópicos en Ciencias de la Computación

Optimización en Sistemas Multi-Agentes



Integrantes:

Natalia Maury  
Joaquín Galván  
Carlos Iparraguirre

Profesor

Willy Ugarte

Ciclo: 2022 - 01

**Introducción**

Los sistemas multi-agentes (MAS = Multi-Agent System) son sistemas que contienen múltiples agentes inteligentes autónomos en un ambiente. Los principios principales de MAS son: autonomía, descentralización y vista local. La autonomía está presente ya que los agentes son conscientes de sus decisiones y la descentralización ocurre al no tener una instancia o elemento que los controle. Además, los agentes pueden ser de diferentes tipos y dependiendo de este es que sus características e impacto variarán. La vista local quiere decir que los agentes por lo general no tienen conocimiento de todo el ambiente porque puede generar soluciones muy complejas. El ambiente donde están los agentes puede ser simulado o virtual, discreto o continuo. Sin embargo, los ambientes también pueden ser clasificados según su accesibilidad, dinámicas, dimensionalidad, etc. (Wikipedia; 2022)

El objetivo del trabajo final es diseñar y aplicar los sistemas multi-agentes (MAS en inglés) para resolver un problema simulado de la vida real en un entorno virtual. En este caso, el problema es una plataforma virtual de delivery que debe organizar a sus repartidores para entregar los pedidos a los ciudadanos de una ciudad. Las personas realizan los pedidos a través de un aplicativo en su dispositivo inteligente, donde realizan un pedido a cierto restaurante. Este aplicativo contiene múltiples restaurantes ya que al ser una ciudad, existen múltiples restaurantes. Si bien existe gran cantidad de restaurantes, las tareas de estos son las mismas: al obtener un pedido a través de la plataforma deben realizar el pedido y luego enviar un repartidor(a) para que lo entreguen a su destino.

Para la resolución del problema, se requiere la implementación de tres tipos de agentes. El primer agente requerido son los ciudadanos de la ciudad que son los que realizan los pedidos a través de la app al restaurante según sus deseos o necesidades. El segundo agente requerido son los restaurantes quienes reciben la orden y la preparan para su delivery. Finalmente, el último agente requerido son los repartidores quienes entregan la orden de los restaurantes a la casa del ciudadano (destino). Cada agente según su tipo tendrá diferentes planes según sus funciones y también una comunicación entre ellos. No obstante, la regularización del comportamiento de las personas puede ser explícita utilizando una especificación organizacional o aleatoria.

El problema tiene algunas condiciones que deben ser satisfechas para tener una implementación exitosa. Una de ellas es la capacidad máxima de pedidos que puede recibir cada restaurante y las reglas generales del problema. La capacidad máxima de pedidos que puede preparar cada restaurante es de 10, es decir, sólo puede tener un máximo de 10 órdenes en preparación pero más de 10 órdenes finalizadas esperando repartidor. Teniendo esto en cuenta, debe haber una interacción entre los ciudadanos y los restaurantes para que las personas puedan realizar los pedidos. Una vez que se terminan las órdenes, un repartidor que esté disponible irá al restaurante a recoger la orden y entregarla a su destino. El repartidor sólo puede entregar 1 orden a la vez. Mientras la orden está siendo preparada y entregada, el ciudadano que realizó el pedido esperará hasta que reciba su pedido. Una vez establecido el proceso, el problema debe ser programado en python con 2400 multi- agentes mínimo: 2000 ciudadanos mínimo, 300 restaurantes mínimo y 100 repartidores mínimo. Finalmente, debe haber una interfaz para mostrar la solución realizada.

**Problema**

Se delimitaron ciertas reglas y conductas para cada agente según su objetivo. Para los repartidores, sólo pueden recoger un pedido a la vez y al buscar un restaurante éste deberá ser el más cercano a su posición. En cuanto a su conducta, tendrán los que siguen las reglas y los que no siguen las reglas. Si alguno que no siguen las reglas es descubierto, estos recibirán la pena de muerte y serán eliminado.

Para los restaurantes, sólo pueden tener 10 órdenes en preparación, pero sus órdenes terminadas podrán variar entre 20 y 30. Una vez llegadas al límite establecido, no se podrán recibir más órdenes. Estos tienen una posición fija, no se moverán en ningún momento, pero su posición será asignada aleatoriamente.

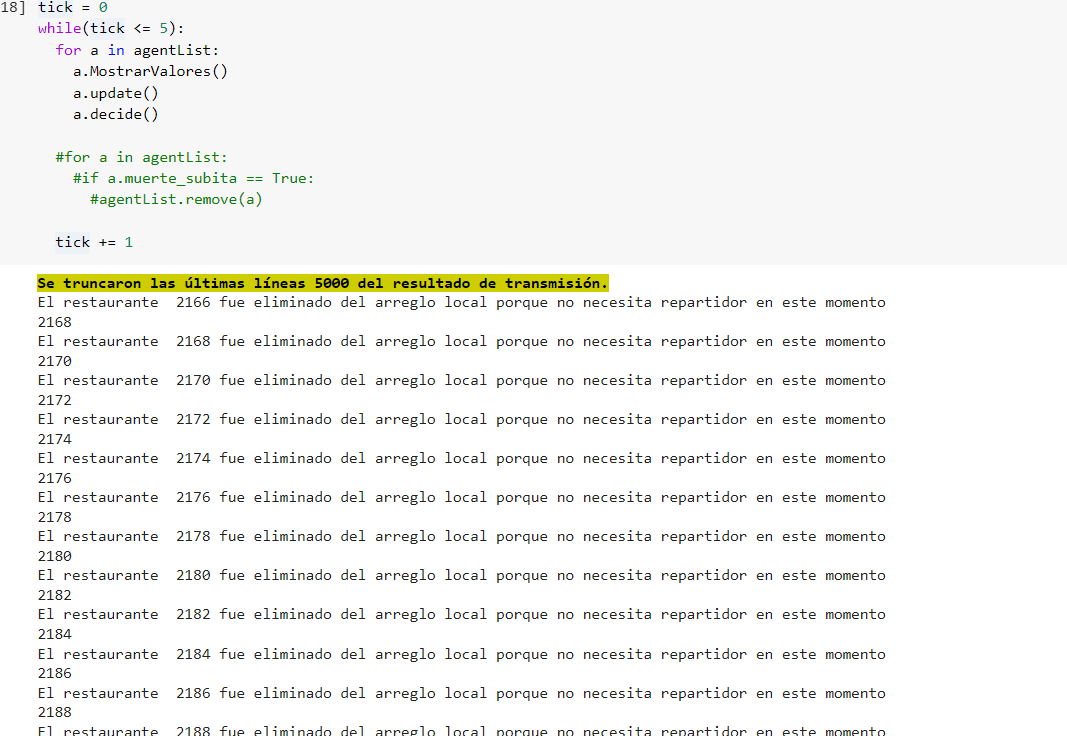
Los ciudadanos sólo pueden realizar un pedido por vez, y hasta que no reciban su pedido no pueden realizar otro. Ellos no se moverán de su posición asignada y la posición será asignada aleatoriamente. En cuanto a su conducta, estos serán divididos en super consumidores quienes realizarán muchos pedidos y con los poco consumidores quienes realizarán pocos pedidos.

En base a esto, se puede determinar el tipo de agente para realizar para cada elemento requerido. Los agentes serán híbridos debido a que reaccionan ante un elemento y en base a esto toman sus objetivos es que tomarán decisiones.

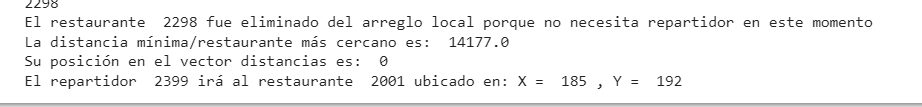
**Resultados**

Primero, el algoritmo creaba los agentes, pero luego de cierto número de iteraciones ocurría un error en el decide() de repartidor. Este error consiste en que la variable h no es asignada correctamente como restaurante en una de las funciones internas del decide(), siendo asignado erróneamente como ciudadano.

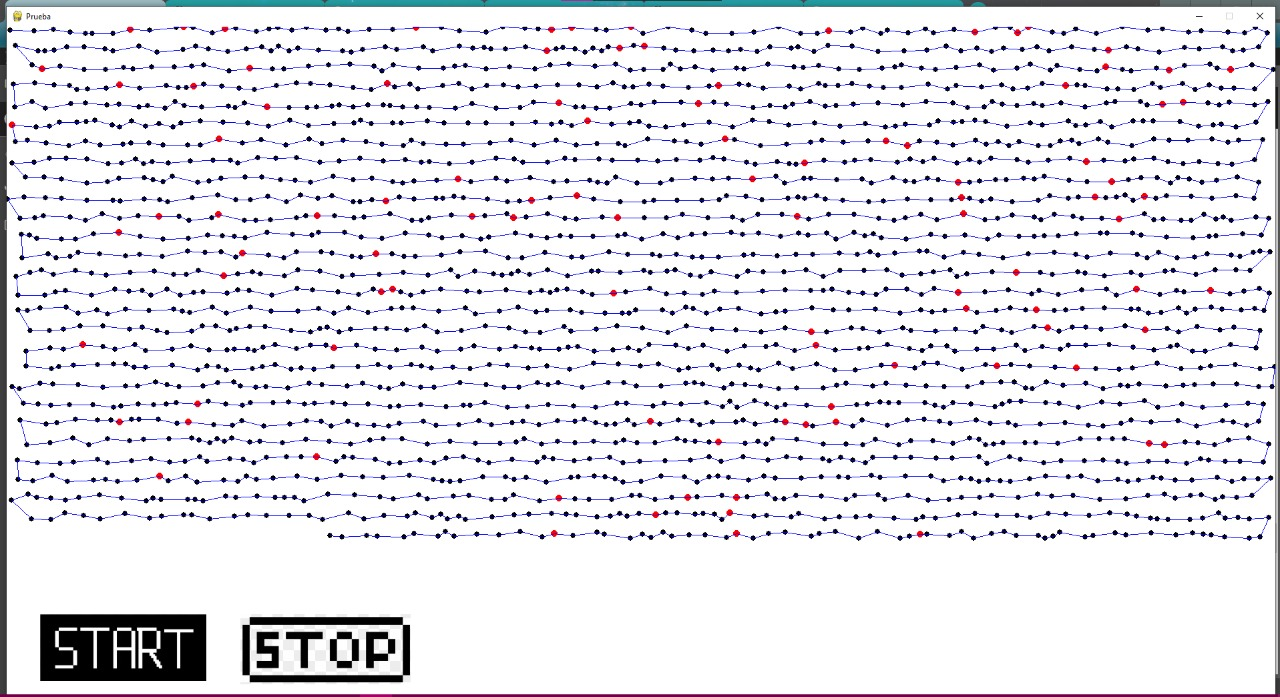
Luego se logró corregir ese error y ahora funciona, pero luego de unos minutos, se truncan las iteraciones y se detiene el algoritmo. Esto ocurre por los prints que se tenía para saber qué salía y si funcionaba el algoritmo. Por lo que se puede observar, sí funciona, pero se deben eliminar los prints para poder ver el resultado final o aumentar el contador y así ver cómo evoluciona el sistema.



En base a lo que se puede leer, se observa que el repartidor logra identificar los restaurantes que necesitan entregar alguna órden, calcular la distancia y saber cómo ir hacia su destino.



En cuanto a la interfaz, esta tiene el control para iniciar la simulación y muestra a todos los agentes en el mapa.



**Conclusiones**

El algoritmo creado funciona parcialmente para resolver el problema. Logra crear los agentes e inicializarlos, también llamar a sus métodos y ejecutarlos. El problema encontrado, es que debido a los prints y por el contador es que no se pueden ver todos los resultados. Es por ello que no se puede dejar corriendo por periodos extendidos de tiempo, ya que no permite mostrar todo el proceso y eventualmente el programa se detiene por todo lo que debe mostrar en pantalla.

La idea era buena ya que aprovecha las cualidades de cada agente y los hace funcionar en conjunto para lograr un objetivo común. Un ejemplo de esto es que los repartidores vayan al restaurante más cercano y no se queden esperando en un sólo restaurante para esperar las órdenes. Esto permite que así se evite en lo posible que los restaurantes se queden con muchas órdenes terminadas.

Link del Collab: <https://colab.research.google.com/drive/1qGJyuWKFVQuZQXDArvLnFjNcguQRnD7S?usp=sharing>

**Bibliografía**

Wikipedia (2022). *Multi-agent system*. Recuperado de <https://en.wikipedia.org/wiki/Multi-agent_system#:~:text=A%20multi%2Dagent%20system%20> [Consultado el 25 de junio del 2022]