

Diciembre 2022



**Tecnológico
de Monterrey**

Reporte de la solución del reto

**Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro**

**Modelación de sistemas multiagentes con gráficas
computacionales**

TC2008B.301

Pedro Oscar Pérez Murueta
Alejandro Fernández
Denisse L. Maldonado Flores

Presenta:

Said Ortigoza | A01707430
Andrés Magaña | A01067963
Ricardo Cáceres | A01706972

Introducción

Los retrasos en el transporte como consecuencia de la congestión (embotellamientos), tienen como consecuencia grandes pérdidas de tiempo y económicas para los habitantes de áreas urbanas y suburbanas. Situación que al mostrar un aumento a nivel global debido a la creciente población y al desarrollo de las zonas urbanas se ha convertido en una situación preocupante y causa de descontento y frustración para los ciudadanos.



Fig. 1: Congestión vehicular cerca del centro de Los Ángeles, CA.

Existen varios tipos de congestión vehicular: recurrente y no recurrente, cuya diferencia principal radica en la previsibilidad de la congestión, a su vez la congestión no recurrente se divide en tres categorías: impredecible, parcialmente predecible y muy predecible.

Para el presente reporte, se simulará una situación de embotellamiento no recurrente impredecible, con el comportamiento que normalmente tendrían los coches que viajan en la vía, para evaluar si nuestra solución presenta o no una mejora para esta situación.

Utilizamos la librería Mesa de Python para modelar el comportamiento de los agentes dentro del modelo y la plataforma de desarrollo Unity para la parte gráfica. Utilizando un arquitectura cliente-servidor interactúan vía POST mediante un servidor de Python y un cliente C#.

Arquitectura del sistema

Nuestras simulaciones computacionales cuentan con una arquitectura multi agente con características específicas tanto para el modelo como para los agentes.

- Agente

En este caso todos los agentes son vehículos de transporte de pasajeros, que no excedan de 5 asientos. Con las siguientes características:

- Viajan a una velocidad constante de 60 km/h desde que comienzan su trayecto.
- Comienzan en uno de tres carriles aleatorios.
- No muestran comportamiento errático, lo que quiere decir que no se cambian de carril a menos que encuentren un problema en el mismo.
- En todo momento, los vehículos reportan y conocen la posición del resto.
- Ningún vehículo puede colisionar con otro.

- Modelo

Nuestro modelo es una vía recta de tres carriles. Con las siguientes características:

- La longitud de la vía es de 1 km.
- No cuenta con accesos laterales de ningún tipo.
- Recibe un número de vehículos que transitan durante la simulación.
- La generación de los agentes es irregular, para modelar un comportamiento más parecido a lo que sucedería en un flujo real de vehículos.

Después de que la mitad de los vehículos hayan sido generados, uno de los del carril central comenzará a frenar hasta detenerse, y permanecerá así hasta el final de la simulación. Es a partir de aquí donde los vehículos se comportan como lo harían normalmente, y después se comportan como nuestra solución propuesta.

Diagrama de interacción del sistema

El protocolo de comunicación de nuestra propuesta consiste en que una vez que un vehículo del carril central frene hasta detenerse y mantenga esa posición, el primer vehículo del carril central que lo vea enviará un mensaje al resto de los vehículos en todos los carriles, comunicando que existe un problema en su ruta, para posteriormente cambiarse a un carril disponible.

Una vez que los vehículos reciban el mensaje, lo propagarán y si se encuentran en la vía central cambiarán de vía a la primera oportunidad.

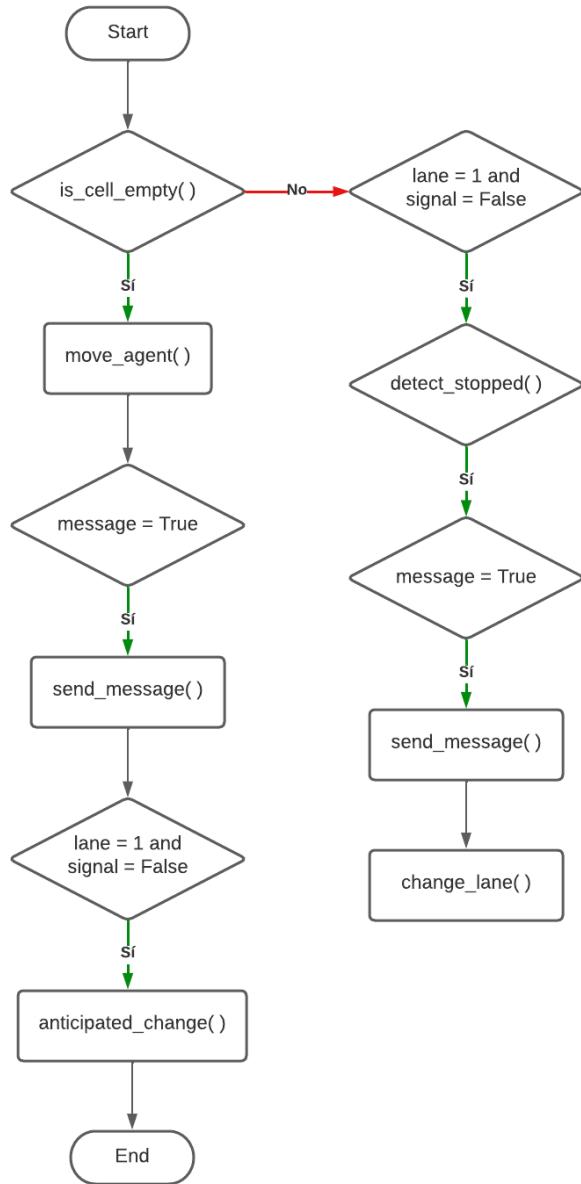


Fig. 2: Diagrama de flujo del sistema

Resultados

Tomando en cuenta que cada vehículo tiene una longitud de cuatro metros, definimos una longitud de 250 celdas para la vía, considerando que tiene que medir 1 km de largo. Definimos una velocidad de cuatro celdas por "step", de manera que con un flujo normal, sin obstrucción en el carril central, le tomaría 62.5 steps a cada vehículo finalizar su recorrido.

```

132  # Definimos las dimensiones del grid.
133  WIDTH = 3
134  HEIGHT = 250
135
136  # Definimos el número de agentes.
137  NUM_CARS = 850

```

Fig. 3: Parámetros utilizados para las simulaciones.

Durante la simulación del comportamiento del sistema en un embotellamiento normal, obtuvimos un promedio de 63.25 steps, que traducidos en segundos, significan un retraso de 0.67 segundos con los parámetros establecidos.

Tiempo promedio de traslado: 63.26148409893993

Fig. 4: Resultados obtenidos durante la primera simulación.

Implementando nuestra propuesta, obtuvimos una mejora del 0.5% aproximadamente, esto debido a que los vehículos toman en promedio 63 steps para finalizar su recorrido. Si bien este número podría parecer poco significativo, modificando la longitud de la vía y aumentando el flujo de agentes, este porcentaje comienza a aumentar, ya que la diferencia principal entre los sistemas es que en la segunda simulación los vehículos tienen una mayor probabilidad de cambiarse a una vía disponible sin perder tiempo detenidos ni reducir su velocidad.

Tiempo promedio de traslado: 63.0942285041225

Fig. 5: Resultados obtenidos durante la simulación que integra la propuesta

Enlace a Github

<https://github.com/AndresMaganaPerez/RetoMovilidadUrbana>

Enlace al vídeo de la solución

https://drive.google.com/file/d/172R06Ex3eoMP5Jj0s3XckqhrrToZzGeA/view?usp=share_link

Reflexiones

- **Said Ortigoza**

Esta Unidad de Formación nos presentó el concepto de sistemas multi agentes y nos introdujo de manera intermedia al manejo de gráficas computacionales utilizando la plataforma Unity, fue interesante ver cómo el desarrollo de simulaciones de estos sistemas puede ayudar a modelar soluciones a problemas complejos del mundo real, en este caso nuestra solución presentó resultados positivos, y podría afirmar que si nuestro sistema se comportara de una manera aún más apegada a la realidad podría tener un beneficio mayor al problema de la congestión vehicular.

Implementar nuestra solución con la tecnología actual no sería demasiado complicado, ya que integrando un mecanismo similar al de nuestra propuesta a una aplicación como Waze podría reducir los tiempos de traslado y los embotellamientos que como mencionamos anteriormente, representan un problema cada vez más común.

Considero que la solución se podría mejorar aún más si modificamos el protocolo de comunicación entre los agentes, de manera que una vez que el coche se detenga, sea el mismo el que envíe el mensaje a los demás, y se propague, y no que sea el coche que lo detecta el que lo haga, ya que aunque este cambio significa una reducción mínima en el tiempo promedio de traslado, a gran escala este número aumentaría significativamente.

La modelación de sistemas multi agentes tiene una amplia variedad de aplicaciones, ya que es fácilmente adaptable a cualquier tipo de sistema del mundo real, un problema que podríamos solucionar aplicando lo aprendido en clase es la urbanización, se podría analizar el comportamiento actual del crecimiento de las ciudades, para observar cómo se comportaría a lo largo del tiempo y analizar si se debe corregir o no mediante cambios en las características y el comportamiento del mismo, ya que un control inadecuado de la urbanización puede tener consecuencias tanto para la sociedad como para el medio ambiente.

- **Andrés Magaña**

Este reto estuvo muy interesante. Mi parte favorita fueron los modelos de python. Ese tema me llamó mucho la atención, puede abordar muchos temas y tiene muchas aplicaciones en la vida real. Por mi parte voy a investigar y abordar este tema. Este tema fue aplicado en temas como covid, propagación de virus, análisis de sistemas.

Nuestro reto sí obtuvo los resultados esperados por parte del reto. La parte de modelado con matplotlib funciona al 100% en las 3 simulaciones. Por parte de Unity no está implementado de la mejor manera. Tuvimos que buscar soluciones rápidas y sencillas de implementar por falta de organización y tiempo con el equipo. Sin embargo, tenemos los conocimientos necesarios para poder implementarlo de la mejor manera que se nos enseñó. Por ejemplo agregar Rigidbody a los objetos para que simulen cómo chocan entre ellos, que desaparezcan los carros al salir de la carretera, que la escala de unity sea proporcional a la escala en python. Además, en Python podríamos generar cambios al planteamiento de los agentes para que pueda graficarse mejor con Unity.

Es factible utilizar la solución con la tecnología actual. Por ejemplo, el primer ejemplo que se me viene a la cabeza es con Teslas. A la hora de utilizar piloto automático e Inteligencia artificial, su comportamiento se vería relacionado con sus alrededores y en el sistema en el que se encuentren. Estas implementaciones de modelo-agente, no es lo único que necesitarían para poder manejar en piloto automático, pero es una parte del sistema para que este funcione en el vehículo.

- **Ricardo Cáceres**

A lo largo de toda la unidad de formación, pude aprender sobre diferentes temas y conceptos como los son los agentes y las gráficas computacionales. Estos temas fueron temas que abrieron la puerta a mi curiosidad e imaginación ya que son temas los cuales vemos en muchas de las tecnologías que ocupamos hoy en día pero no nos damos cuenta. La modelación de agentes pienso que es algo de muchísima utilidad ya que esto nos puede ayudar a modelar, entender y proponer soluciones a problemáticas de la vida real.

Para el reto de esta unidad de formación, se nos pedía simular una congestión vehicular y brindar una solución a través del modelado de los agentes y graficando esta simulación en 3d con ayuda de Unity.

Nuestra solución dada si se podría implementar en tecnologías actuales como son las aplicaciones de tráfico. Estas aplicaciones muestran un “warning” de embotellamiento en la ruta. Y nuestra solución hace justamente eso. La comunicación de agentes permite que los demás agentes sepan que hay un embotellamiento desde antes que lleguen al punto del embotellamiento. Nuestra solución por parte del modelado en Python con la ayuda de la librería de Mesa funciona muy bien. Sin embargo, en la graficación de Unity siento que si pueden haber mejoras. Por ejemplo, una mejora que puede haber es que los coches tengan un RigidBody para que si llegan a chocar, de verdad se muestre la colisión en la simulación. De la misma manera, me hubiera gustado que los coches al cambiar de carril se miraran de forma natural. Y que las dimensiones del modelo en Unity fueran de mejor escala. Pero en general puedo decir que nuestra solución dada funciona bastante bien.

Siento que los temas aprendidos en esta unidad de formación son aplicables en muchos ámbitos de la vida real. En ámbitos más recientes como lo son los coches autónomos. Estos coches deben de tener la capacidad de decidir acciones de acuerdo al ambiente en que se encuentren. Justo como los agentes implementados en nuestra solución.