Simulación de tráfico vehicular

Érick Ostorga[1] y Erick Palma[2]
CI-1441 Paradigmas computacionales
Escuela de Ciencias de la Computación e Informática
Facultad de Ingeniería
Universidad de Costa Rica
[1] *ostorgach.erick@gmail.com*, [2] *erickpalma.solano@gmail.com*

Julio de 2015

Resumen

El problema tratado en la aplicación de simulación de tráfico en un ambiente multiagente es basado en un modelo de la librería de NetLogo llamado Traffic Grid el cual contempla una cuadrícula de automóviles con semáforos que cambian cada cierta cantidad de ticks. La mejora que se propone en este modelo es hacer que los automóviles posean velocidad variable, y que haya semáforos inteligentes, como también eliminar el comportamiento circular de los vehículos en la cuadrícula y comparar los resultados obtenidos en ambos modelos.

Palabras clave: Sistema multiagente, tráfico vehicular, semáforo inteligente, NetLogo.

1. Introducción

El tráfico vehicular es un aspecto de la vida que, hoy en día tiende a aumentar y a dificultarse por la cantidad de vehículos que se encuentran en las autopistas. Por lo que se pensó en un problema que se pueda resolver y analizar junto con el paradigma de sistemas multiagentes. Se tiene una cuadrícula basada en un modelo de NetLogo llamado Traffic Grid el cual contiene una simulación de movimiento vehícular junto con semáforos que cambian de señal cada cierta cantidad de *ticks* de máquina.

En este modelo no hay inteligencia alguna, pues los automóviles se mueven a una velocidad constante y frenan de golpe al encontrarse otro en frente ó el semáforo en rojo. Además la cuadrícula presenta un comportamiento circular, por lo que los carros que salen por un extremo entran de nuevo por el lado contrario. Debido a esto es que se desea plantear una solución.

La solución al Traffic Grid consiste en analizar este comportamiento y realizar ajustes en cuanto al diseño del mismo. Se planeó cambiar la forma en que los vehículos se les asigna la velocidad, por lo que puedan ir frenando desde antes de chocar con el que tienen en frente, un movimiento más acertado a la vida real. Por otro lado se elimina el aspecto circular para poder tener diferente cantidad de carros en cada carril y no siempre la misma. Por último se agrega una inteligencia a los semáforos que depende de la congestión vehicular que tengan en ese carril deciden cambiar a verde.

2. Marco teórico

2.1. Enfoque de aprendizaje y conocimiento de agentes

Cuando se habla de sistemas multiagentes es útil recordar que una de las características de los mismos es saber comportarse en diferentes entornos y poder

actuar por sí mismos, por lo que el enfoque de aprendizaje obtenido por medio del ambiente y conocimiento de lo que está pasando es uno de los aspectos más importantes para mostrar resultados coherentes en el proyecto.

Para este estudio se refiere a (Cuena et al, 1999) donde se intenta resolver el problema de manejo de tráfico inteligente basado en conocimiento usando un modelo TRYS (IDE de modelaje semejante a NetLogo). En esta publicación se propone un modelo de estructura física, donde se manejan problemas del tráfico y el control del mismo. La utilidad de esto para el proyecto es que al ser basado en conocimiento, es un tema que se vive a diario y ayuda a orientar el modelo hacia una perspectiva más real, pero también se limita a que debido al acercamiento a la realidad pueden haber cambios de leyes de tránsito u otros temas que involucren a los agentes a cambiar de aprendizaje de un momento a otro, por lo que se torna más complicado.

2.2. Enfoque de técnicas orientadas a agentes

Para aprender un poco sobre cómo asignarles distintas funciones y técnicas a los agentes es importante pensar que esto ayuda a mejorar la forma en que se implementaría la manera de mejorar el tráfico eficientemente. Y estas técnicas estarían asociadas al conocimiento que tiene cada agente en su entorno.

Para este enfoque se refirió a (Burmeister et al, 1997) donde intentan mejorar el tráfico y el transporte de una manera ecológica y eficiente, todo esto con técnicas orientadas a agentes (AOTs). También proponen diferentes consejos a realizar en un sistema multiagente inteligente el cual se puede asociar a la vida real para la mejora del tráfico. Se propone en su publicación una gran cantidad de opciones para un flujo del tráfico más cómodo y ordenado, pero se limita a que unas personas no pueden realizar tanto cambio por ellas mismas, puesto que los gobiernos tienen mucha fuerza en ese aspecto.

2.3. Enfoque de autonomía en agentes

La autonomía en los sistemas multiagentes es de los aspectos más vitales para el buen funcionamiento de cualquier modelo basado en este paradigma. Y es que el hecho de que un agente sepa comportarse en un ambiente cambiante es de gran utilidad para acercarse al hecho de enfrentarse a un paradigma de inteligencia artificial por otro lado.

Para esto se enfocó en (Roozemond, 1996) quien intenta demostrar que el manejo de tráfico dinámico no es 100 % ejecutable y que se requiere de demasiadas pruebas para su buen funcionamiento, lo demuestra haciendo un análisis de sistemas multiagentes autónomos, el autor propone un sistema dinámico de manejo de tráfico en ambientes cambiantes, lo cual se acerca al modelo propuesto en este documento.

Una gran ventaja de artículos como esto es que son sistemas con inteligencia artificial y que aprenden de la actualidad, mientras que es limitado por la cantidad de pruebas que hay que hacer para demostrar su correcto funcionamiento acercado a la realidad.

2.4. Otros enfoques

Hubo revisión de otras referencias bibliográficas que no se acercaron en su totalidad al avance de este proyecto, sin embargo es importante recalcar una de ellas, por parte de (Chen et al, 2008) que es un vistazo sobre aplicaciones multiagentes y móviles en donde se propone una tecnología que pueda mejorar la habilidad de administración del tráfico en un ambiente dinámico. Todo esto con una aplicacion llamada Mobile-C que detecta en tiempo real el tráfico y tiene un sistema de administración.

Se pensó usar este enfoque y otros como ayuda complementaria pero se alejaba un poco de los objetivos específicos y general del proyecto, por lo que se dejaron, por decirlo así, fuera del proyecto.

3. El problema

Se requiere tener un control del tráfico vehicular dentro de una cuadrícula de un tamaño específico, ya que esto ayudaría a comprender el problema de una

mejor manera para analizar de manera visual lo que está ocurriendo y así descifrar sus posibles soluciones.

Se desea tener un cambio en la inteligencia de semáforos y vehículos, asignarles diferente tipo de funciones para que se comporten como simular un sistema multiagente donde todos interactúen entre sí y sepan cómo reaccionar ante distintas situaciones.

4. Objetivos y cronograma

4.1. Objetivo general

- Diseñar e implementar un modelo en el lenguaje de programación NetLogo que genere un proceso automatizado de tráfico vehicular en una cuadrícula, enfatizado en la mejora del tiempo de espera de los vehículos en la cuadrícula y el tiempo de estar detenidos.

4.2. Objetivos específicos

- Controlar la velocidad vehicular, agregando desaceleración y aceleración variable.
- Controlar la sincronización de los semáforos, teniendo control de la cantidad de vehículos por parcela.
- Manejar creación de automóviles aleatoria eliminando así el comportamiento circular.
- Lograr que los vehículos puedan doblar en las intersecciones.

4.3. Cronograma

La planificación realizada para el cumplimiento del proyecto y sus objetivos se muestran en la Tabla 1.

5. La propuesta de solución

5.1. Conceptualización de agentes y entorno del modelo

El modelo que se propone implementar está basado en el paradigma computacional denominado Sistema multiagente (SMA), por este motivo es necesario determinar cuáles han de ser los componentes y sus características.

- Agentes
  - Vehículos: se modela un solo tipo de vehículo, automóvil liviano, el cual realiza las acciones de aumentar y reducir su velocidad dadas las opciones de acelerar y desacelerar, frenar completamente al tener otro vehículo frente a él o en la luz roja de un semáforo, y por último, cambiar de dirección a otra calle en una intersección.
  - Semáforos: cada semáforo independiente se idealiza como un agente que realiza la acción de contar una cantidad definida de vehículos detenidos frente a él, de manera que pueda determinar si cambia su estado (de luz verde a luz roja).
- Entorno: se define el entorno del modelo como una cuadrícula de tamaño tres cuadras por tres cuadras (3×3) que representa un mapa vial de una ciudad, el cual contiene nueve intersecciones cada una con un semáforo. Cada una de las calles tiene un carril de circulación en el que se permite el tránsito de vehículos en un solo sentido de vía.

5.2. Estrategias para la solución de los objetivos del modelo

5.2.1. Propuesta para crear vehículos aleatorios

Se propone manejar los límites de la cuadrícula en los ejes X y Y para poder controlar cuando cada vehículo va pasando por ahí, simular que sale de la autopista matando al agente y creando otro nuevo en cualquier otro carril de la cuadrícula. Es decir se elimina el comportamiento circular del modelo y se puede tener aleatoria cantidad de carros en cada carretera, produciendo así posibles congestiones vehiculares que es lo que se quiere probar para tener control de los semáforos inteligentes.

Tabla 1: Planificación del diseño e implementación del modelo

Semana	Objetivo	Descripcion de trabajo
17/04 al 01/05	Definir el tema investigación	Planteamiento del tema y problema, objetivos y revisión de la bibliografía existente
04/05 al 08/05	Revisar más profundamente la bibliografía y entender las características del paradigma	Completar el marco teórico e iniciar la representación y conceptualización del problema
11/05 al 15/05	Aprender el lenguaje de programación a utilizar	Revisión de modelos existentes y realización de tutoriales sobre NetLogo.
18/05 al 22/05	Redactar secciones del informe escrito	Continuar la redacción de las secciones empezadas del informe escrito
25/05 al 29/05	Empezar a expandir el modelo base de NetLogo	Agregar las primeras funcionalidades
01/06 al 05/06	Afinar detalles del informe escrito	Redacción de secciones del informe escrito (a excepción de desarrollo y resultados)
08/06 al 12/06	Afinar detalles del modelo programado	Continuar la expansión del modelo programado
15/06 al 19/06	Afinar detalles del modelo programado	Continuar la expansión del modelo programado
22/06 al 26/06	Afinar detalles del modelo programado	Continuar la expansión del modelo programado
30/06 al 03/07	Completar el modelo	Terminar el modelo programado
06/07 al 10/07	Completar el proyecto	Presentación del resultado del modelo al profesor

5.2.2. Propuesta para manejar la velocidad variable

Al tener modelo un solo tipo de vehículo, se desea que cada uno de los agentes que se produzcan tengan velocidad variable y de esta manera aportar un mayor realismo a la simulación del tráfico vehícular. Esto se propone lograr asignando un valor aleatorio -en tiempo de ejecución- al valor total de avance del vehículo.

5.2.3. Propuesta para semáforos inteligentes

Para poder simular semáforos inteligentes que intenten descongestionar el tráfico detenido en rojo se propone hacer un conteo de los vehículos que están detenidos en frente del mismo y así cambiar de color a verde según la cantidad de carros que el usuario desee poner como máximo para cambiar de color en el semáforo.

6. Desarrollo, prueba y validación

Para desarrollar la solución a los objetivos planteados se debía empezar por aprender el lenaguaje de programación NetLogo, ya que no se tenía conocimiento previo de este. Una vez que se logró dominar la base del lenguaje, se debía analizar y entender el modelo Traffic Grid, el cual era la base de la cual se partiría para desarrollar el modelo propuesto. En una primera etapa de mejoró una de las limitaciones más importantes del modelo existente: agregar la funcionalidad de que cada vehículo pudiera tener una velocidad variable.

Un problema inicial encontrado con esta mejora fue que al asignar un número aleatorio al valor que permite avanzar al vehículo, este en algunos omitía la señal de un semáforo y seguía su camino. Esto se debía, a grandes rasgos, a que la forma en que un vehículo decide detenerse en un semáforo en rojo es poner su velocidad nula (en '0'), pero como el valor que determina cuánto avanza un vehículo es la velocidad que posee en ese momento, al agregarle un valor aleatorio su velocidad ya no sería nula, sino el valor aleatorio. La solución a este problema fue determinar si el vehículo se encontraba en un semáforo o no, si no era el caso, entonces sí se le asignaba un valor aleatorio a su velocidad.

El siguiente paso fue agregar las funcionalidades de acelerar y desacelerar. Para esto, se tomó como base otro modelo existente en la biblioteca de NetLogo llamado Traffic Basic, el cual implementa un flujo simple de vehículos en una vía. Este modelo sí incluye las funcionalidades deseadas, sin embargo, la forma en que las implementa debían acoplarse de forma distinta en el modelo propuesto.

Luego de acoplar estas funcionalidades se presentó el problema que, visualmente, solo se lograba apreciar el cambio progresivo de la aceleración de un vehículo pero no el de la desaceleración. La razón del problema era que la aplicación del valor de desaceleración a la velocidad del vehículo se daba en el momento en el que el vehículo tenía a otro justamente al frente, esto unido al control que se tiene de frenar por completo un vehículo antes de entrar a la parcela de otro, es decir, evitar un choque. La solución fue que el vehículo que debía disminuir la velocidad lo hacía de manera adelantada contando una cantidad específica de parcelas adelante y determinar si había otro vehículo; si se da el caso divide su propia velocidad a la mitad y le disminuye el valor de la desaceleración. La división de la velocidad a la mitad se decidió por motivos más visuales ya que los valores de aceleración y desaceleración debían ser pequeños, por lo que seguía sin notarse tanto.

En el momento de implementar la característica de que los vehículos salieran por completo de la cuadrícula se presentó el problema que no solo el vehículo que se encontraba en uno de los límites se eliminaba, sino que todos los vehículos que estuvieran en la misma calle a lo largo del entorno también desaparecían. Esto se debía a que la forma de eliminar vehículos era determinar cuando uno se encontraba en uno de los límites, y por error, preguntaba a todos los agentes en la misma línea que se eliminaran. La solución fue ejecutar la acción de eliminar directamente en el agente que cumpliera las condiciones.

El siguiente problema que se presentó fue al querer agregar más carros luego de que uno saliera de la cuadrícula. Inicialmente, no se especificó que el nue-

vo vehículo debía crearse justamente en el inicio de alguno de los ejes, por este motivo, el nuevo vehículo podía aparecer a la mitad de una calle. Lograr el objetivo de que los semáforos cambiaran la señal al tener una cantidad específica de vehículos esperando fue relativamente fácil. Sin embargo, la lógica utilizada en un inicio no fue la mejor. Primero, se cambiaba la señal del semáforo cuando este encontraba un vehículo detenido en la parcela en la posición que determina el límite de vehículos a esperar, pero ya que algunos vehículos guardan distancia entre otros para evitar choques no necesariamente había la cantidad de vehículos deseados esperando. Por ejemplo, el semáforo identificaba si cinco parcelas frente a él había un vehículo detenido, si lo había entonces cambiaba su señal. Pero, podía ser el caso que un vehículo -entre esa cantidad de parcelas- guardó una distancia de casi una parcela respecto al vehículo en frente, por lo que no se cumplía que en cada una de las parcelas había un vehículo detenido, así, la señal cambiaba con cinco o menos vehículos. La solución a esto fue implementar la lógica adecuada, que el semáforo contara inmediatamente desde él cuántos vehículos con velocidad cero había, si contaba el número límite determinado cambia la señal.

Otros problemas que quedaron sin poder solucionarse al final de este proyecto se presentan en la sección 8 de este documento.

7. Experimentación y análisis

Para poder experimentar con el modelo y analizar sus resultados, NetLogo brinda unas herramientas muy útiles para obtener información del modelo que se está corriendo, entre ellos encontramos gráficos que nos ayudan a medir la cantidad de tiempo que los vehículos pasan detenidos, la velocidad promedio que se tiene en el modelo según la velocidad de todos los carros en cada *tick* y también la cantidad de carros que hay detenidos en cierto momento.

Gracias a estos gráficos se puede experimentar los resultados con los cambios realizados en el proyecto. Además, para analizar estos resultados se va a procesar tanto el modelo realizado por NetLogo y el propuesto, es decir, Traffic Grid y el modelo creado en este proyecto. Así, se tomarían dos diferentes resultados y se podría comprobar que éste último tiene una mejoría sobre el de la biblioteca de NetLogo.

Por parte, del modelo de Traffic Grid, cuyos resultados se presentan en la Imagen 1, se puede observar que con 50 vehículos con velocidad límite de 1 en una cuadrícula 3×3 el promedio de carros detenidos en 1000 *ticks* ronda por los 25-28, la velocidad promedio es de 0,5 y el tiempo de espera promedio de los vehículos es de 10,2 *ticks*.

Al correr el nuevo modelo con los mismos datos de prueba del Traffic Grid: 50 vehículos con velocidad límite de 1, pero con las características propias de este modelo: aceleración y desaceleración en 0,009, y con una cantidad de espera de cinco vehículos en semáforo se obtienen los siguientes resultados: para un tiempo de 1000 *ticks* un promedio de 20 vehículos detenidos, la velocidad siempre es menor a un quinto de la velocidad máxima. Finalmente, un tiempo de espera en promedio de 40 *ticks*, con un pico máximo de 77,2 *ticks*. Los gráficos completos se muestran en la Imagen 2.

Con estos datos mostrados en los gráficos se puede observar que el modelo propuesto obtiene valores más oscilantes que el original, probando que está más cercano a la realidad, en cuanto a que las condiciones no siempre son las mismas.

Ahora, se puede correr el mismo modelo nuevamente con las mismas condiciones iniciales de la prueba anterior para observar que los resultados no siempre son los mismos, pero sí muy parecidos. Es decir, al haber escenarios diferentes en cada corrida, el sistema de semáforos logra mantener una constancia entre la cantidad de vehículos detenidos y su tiempo de espera. Los gráficos para esta segunda prueba se muestran en la Imagen 3. Para un tiempo de 1000 *ticks* se tiene un promedio de 30 vehículos detenidos, la velocidad sigue siendo menor a un quinto de la velocidad máxima. Finalmente, un tiempo de espera aproximada de 40 *ticks*, con un pico máximo de 71,3 *ticks*.

El modelo propuesto posee dos formas de realizar la sincronización de semáforos. En la primera, la cual tiene la opción de *ticks* deshabilitada, realiza el cambio de señal cada vez que se encuentra una cantidad de vehículos detenidos frente a un semáforo; la segun-

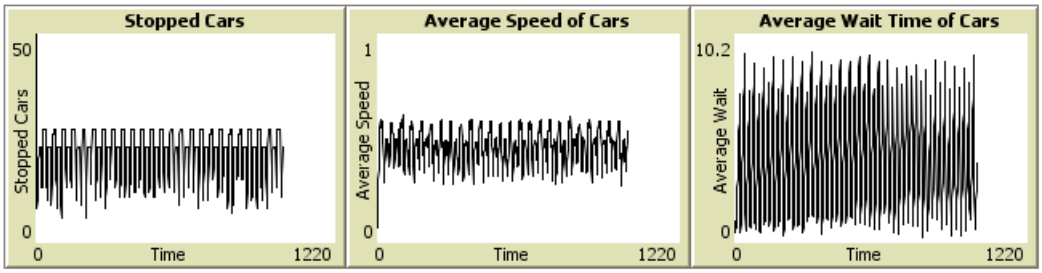

Imagen 1: Gráficos modelo Traffic Grid

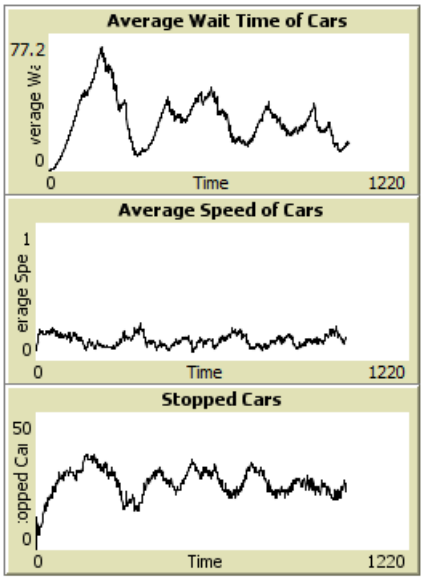

Imagen 2: Opción 1 de sincronización. Corrida 1

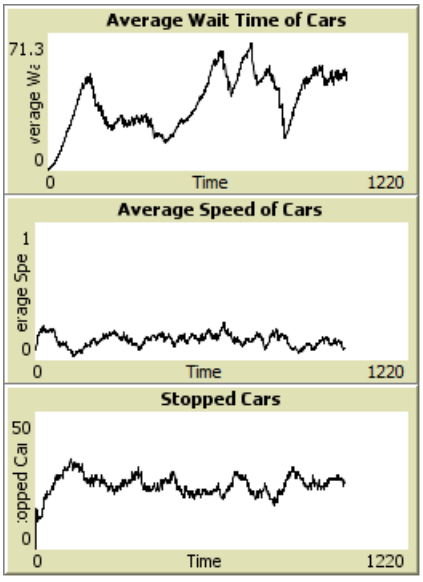

Imagen 3: Opción 1 de sincronización. Corrida 2

da, además de estar revisando que haya esa cantidad de vehículos, cambia la señal de los semáforos en un intervalo de tiempo definido por la cantidad de *ticks* por ciclo del modelo. O sea, sería de esperar que la segunda opción reduzca el tiempo de espera en un semáforo, ya que un vehículo no tendría que esperar siempre que llegue una cantidad específica de vehículos al mismo semáforo -condición que tal vez nunca podría darse o podría tardar mucho-, sino que se le permitiría avanzar luego de un tiempo. Las dos pruebas anteriores (Imagen 2 e Imagen 3) corresponden a la primera opción de sincronización. Para la segunda se muestran a continuación dos pruebas.

La Imagen 4 muestra los resultados al correr el modelo con la segunda opción de sincronización. Para un tiempo de 1000 *ticks*, 50 vehículos con velocidad límite de 1, aceleración y desaceleración en 0,009, una cantidad de espera de cinco vehículos en semáforo y cambio de la señal de los semáforos cada 20 *ticks*, la cantidad de vehículos detenidos disminuye a un máximo de 25 vehículos, con un promedio de 20 vehículos, la velocidad en esta ocasión excede un quinto de la velocidad máxima, pero se mantiene baja. Finalmente, un tiempo de espera apróximado de 15 *ticks*, con un

pico máximo de 22,6 *ticks*.

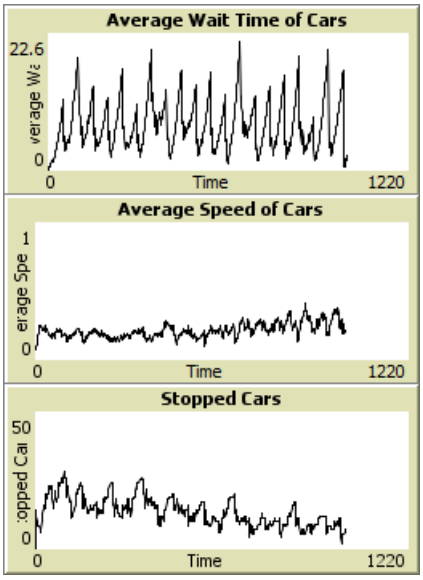

Imagen 4: Opción 2 de sincronización. Corrida 1

Una segunda corrida del mismo modelo anterior con las mismas condiciones iniciales da los resultados mostrados en la Imagen 5. La cantidad de vehículos detenidos tiene un máximo de 28 vehículos, con un promedio de 22 vehículos, la velocidad en esta ocasión excede un quinto de la velocidad máxima, pero se mantiene igualmente baja con un máximo de 0,35. Por último, un tiempo de espera aproximado de 12 *ticks*, con un pico máximo de 23,8 *ticks*.

Como era de esperarse, el modelo propuesto con la segunda opción de sincronización disminuye el tiempo de espera y la cantidad de vehículos detenidos en el tiempo en comparación con la primera opción.

Con las últimas dos pruebas, a simple vista, sería de esperar que el tiempo máximo de espera no sobrepase el tiempo que tarda un semáforo en cambiar de señal, que es de 20 *ticks*. Sin embargo, hay que notar que al haber una fila de vehículos estacionados donde cada uno tiene una aceleración propia, a partir del segundo cada uno de los vehículos en la fila deberá esperar a que el que tiene en frente empiece a avanzar, por lo que probablemente sobrepase el tiempo de espera de cada semáforo.

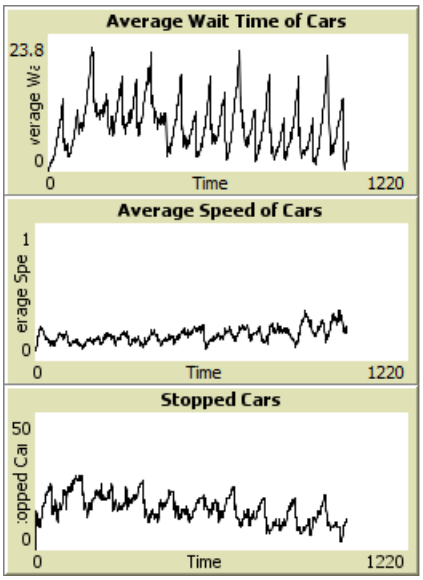

Imagen 5: Opción 2 de sincronización. Corrida 2

8. Problemas abiertos y problemas futuros

8.1. Problemas abiertos

Se generan semáforos intermitentes (cambiando de rojo a verde muy rápidamente) cuando ambos ejes (X y Y) tienen la cantidad necesaria para poder cambiar de color su semáforo respectivo, entonces ambos al cumplir esta propiedad empiezan a cambiar rápidamente hasta que uno de los dos automóviles pase primero y el otro le dé prioridad para pasar.

Se da el caso de que algunos carros que van manejando por los bordes de la cuadrícula y doblan hacia la salida de la misma vuelven a entrar por el extremo contrario y no son eliminados como debería de ser. Este problema no se entiende por qué a veces parece haber sido arreglado y otras veces no.

8.2. Problemas futuros

Si se modifica el programa insertando gran cantidad de vehículos a la cuadrícula, se puede deteriorar gravemente el modelo hasta el punto de que se ter-

mine la aplicación a causa de un error por falta de lugar para crear o mover vehículos.

Si se incrementa el tamaño de la cuadrícula puede darse un problema de creación y eliminación de vehículos, pues el modelo está implementado para funcionar a la perfección con las medidas que se tienen contempladas (3×3).

9. Agradecimientos

Se le agradece al profesor Dr. Álvaro de la Ossa Osegueda por la ayuda constante desde el inicio del curso de Paradigmas computacionales, por brindarnos ayudas bibliográficas, ayudas teóricas y técnicas para poder concluir el proyecto de la mejor manera, logrando comprender el problema y cumplir los objetivos principales del proyecto.

Referencias

B. Burmeister, A. Haddadi y G. Matylis. *Application of multi-agent systems in traffic and transportation*, volumen 144. IEE Proc.-Softw. Eng., 1997.

B. Chen, H. Cheng y J. Palen. *Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems.* 2008.

J. Cuena, J. Hernández y M. Molina. *Real-Time traffic management through knowledge-based models: The Trys Approach.* Boadilla del Monte, Madrid, España, 1999.

D. Roozemond. *Intelligent traffic management and urban traffic control based on autonomous objects.* Departamento de Ingeniería Civil, Países Bajos, 1996.