Proyecto Final Inteligencia Artificial - Simulación

Traffic Lights Optimization

Integrantes:
Leandro Rodríquez Llosa
Laura V. Riera Pérez
Marcos M. Tirador del Riego

Grupo: C-311

Tercer año. Ciencias de la Computación.
Facultad de Matemática y Computación, Universidad de La Habana, Cuba

Enero 2023

I. Repositorio del proyecto

https://github.com/science-engineering-art/traffic-lights

II. Descripción

En todo el mundo, la congestión del tráfico sigue siendo un problema importante en la mayoría de las ciudades, debido al creciente número de vehículos privados, de mercancías y de transporte público. Este fenómeno afecta, sobre todo en horas pico, a los usuarios de la red vial, los cuales pierden mucho tiempo en la carretera; además de incidir de manera negativa en el medio ambiente pues los carros se encuentran más tiempo encendidos liberando gases a la atmósfera.

Se puede pensar en varias soluciones para este problema:

1. Construcción de nuevas carreteras: Muchas veces esto no es posible debido a las condiciones geográficas, y más importante aún, es muy costoso, por lo que en general no es una solución viable.

2. Mejora del sistema de señalización vial: Es más sensata pues se relaciona inteligentemente con la infraestructura existente. Es de especial interés la mejora de los semáforos ya que estos controlan el flujo de la red vial de la ciudad. En estos podemos tener:

   - Plan de luces fijo (Estático): se fijan los tiempos de verde y rojo en cada línea de luces de una intersección así como su secuencia una sola vez teniendo en cuenta las previsiones de tráfico, y estas no cambian.
   - Controladores de tiempo real (Dinámicos): en técnicas de tiempo real, el sistema debe ser capaz de adaptarse inmediatamente (o muy brevemente) a las condiciones del tráfico. Dicho sistema posee algoritmos que permiten controlar el tráfico, los cuales reciben información sobre el estado del tráfico, que ha sido recolectada por los sensores colocados en cada carril, y reacalculan la duración y la sincronización de la luces para minimizar la congestión, es decir, para minimizar el tiempo promedio de espera en las luces, y la duración de colas.

1. Objetivo

Creación de un algoritmo de control que determine de la secuencia de fases y el tiempo de de luz verde óptimos en los semáforos de las intersecciones, con el fin de hacer más fluido el tráfico y minimizar las colas.

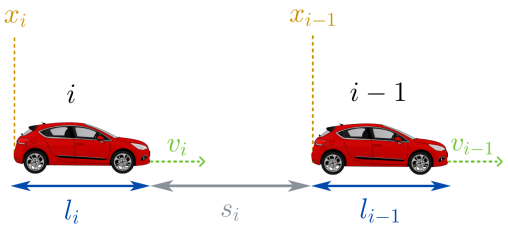
III. Simulación

I. Modelo mesoscópico

Para modelar el flujo del tráfico se utiliza un modelo mesoscópico, modelo híbrido que combina las características de los modelos microscópico y macroscópico.

Como en el modelo microscópico, se representan los vehículos de forma independiente y se intenta replicar el comportamiento de un conductor. En consecuencia, es un sistema multiagente, es decir, cada vehículo opera por sí mismo utilizando información de su entorno.

Cada vehículo está identificado por un número i. El i-ésimo vehículo sigue al (i-1)-ésimo vehículo. Para el i-ésimo vehículo, se denota por x_i su posición a lo largo del camino, v_i su velocidad y l_i su longitud. Sean, además, s_i la distancia parachoques a parachoques y Δv_i la diferencia de velocidad entre el i-ésimo vehículo y el vehículo que le precede (vehículo número i-1).



$$s_i = x_i - x_{i-1} - l_i$$
$$\Delta v_i = v_i - v_{i-1}$$

Por otro lado, como característica del modelo macroscópico, se tienen en cuenta factores globales que describen el movimiento de vehículos como un todo, en términos de densidad de tráfico (vehículos por km) y flujo de tráfico (vehículos por minuto).

revisar y arreglar como se utiliza el modelo macroscopico

II. Modelo Conductor Inteligente

El Modelo de Conductor Inteligente describe la aceleración del i-ésimo vehículo en función de sus variables y las del vehículo que le precede. La ecuación dinámica se define como:

$$\frac{dv_i}{dt} = a_i \left(1 - \left(\frac{v_i}{v_{0,i}}\right)^{\delta} - \left(\frac{s^*(v_i, \Delta v_i)}{s_i}\right)^2\right)$$

$$s^*(v_i, \Delta v_i) = s_{0,i} + v_i T_i + \frac{(v_i \Delta v_i)}{\sqrt{(2 a_i b_i)}}$$

donde:

- $s_{0,i}$: es la distancia mínima deseada entre el vehículo i e i-1.
- $v_{0,i}$: es la velocidad máxima deseada del vehículo i.
- δ : es el exponente de la aceleración y controla la "suavidad" de la aceleración.
- T_i : es el tiempo de reacción del conductor del i-ésimo vehículo.
- a_i : es la aceleración máxima del vehículo i.
- b_i : es la desaceleración cómoda para el vehículo i.
- s^* : es la distancia real deseada entre el vehículo i e i-1.

Interpretando los términos en s^*:

- $v_i T_i$: es la distancia de seguridad del tiempo de reacción. Es la distancia que recorre el vehículo antes de que el conductor reaccione (frene). Dado que la velocidad es la distancia entre el tiempo, la distancia es la velocidad por el tiempo.
- $(v_i \Delta v_i) / \sqrt{(2 a_i b_i)}$: es una distancia de seguridad basada en la diferencia de velocidad. Representa la distancia que tardará el vehículo en reducir la velocidad (sin chocar con el vehículo de delante), sin frenar demasiado (la deceleración debe ser inferior a b_i).

III. Modelo de red vial de tráfico

Para modelar la red vial se utiliza un grafo dirigido, donde las aristas representan las calles (cuadras) y los vértices las intersecciones. Cada vehículo tiene un camino que consta de múltiples calles. Se aplica el Modelo Conductor Inteligente para vehículos en la misma calle. Cuando un vehículo llega al final de la calle, se retira de la misma y es añadido a la siguiente.

IV. Generación de vehículos

Los vehículos se generan en los extremos del mapa. Cada calle tiene un λ asignado que representa la cantidad de carros por segundo que pasan por ella, el cual corresponde a un valor entre 70 y 150 carros por hora. A las calles que se determinan como principales se les aumenta su λ.

Se usa λ para saber la probabilidad de que se genere un carro en dicha calle en un intervalo de tiempo, utilizándolo como parámetro en la función de probabilidad de Poisson. Esta función dirá cuál es la probabilidad de que hayan pasado x carros en un intervalo de tiempo en esa calle.

$$p(x) = f(x, \lambda) = \frac{\lambda^x \cdot e^{-\lambda}}{k!}$$

Luego, se genera un número r aleatorio entre 0 y 1, y se halla la probabilidad $p(0)$ de que no se haya creado ningún carro en este intervalo de tiempo. Se genera un carro en esta calle si $r > p(0)$.

Al generar un carro se establece su destino y su ruta. El destino se selecciona aleatoriamente de acuerdo a una probabilidad que tiene cada calle de que un carro vaya hacia ella, asignándole mayor peso a las calles más transitadas. Cuando se ejecuta por primera vez una simulación en un mapa, se utiliza Floyd-Warshall para hallar los caminos de menor distancia entre todo par de calles, y se guardan los mismos. Luego, se escoge la ruta de estos caminos precalculados.

V. Semáforos

v.1. Turnos

Un turno es un conjunto de semáforos que se encuentran en verde en un mismo período de tiempo. Cada semáforo de la intersección puede existir en más de un turno, siendo posible así establecer distintas combinaciones para las direcciones factibles (seguir recto, doblar a la derecha o doblar a la izquierda) tal que no haya choques y haya la menor cantidad de turnos posibles.

v.2. Zonas

Los semáforos tienen dos zonas en los que los vehículos se comportan de forma diferente:

- Zona de ralentización: Zona en la que los vehículos reducen su velocidad máxima utilizando un factor de ralentización.

$$v_{0,i} := \alpha v_{0,i} \text{ donde } \alpha < 1$$

- Zona de parada: Zona en la que se detienen los vehículos. Esto se logra utilizando una fuerza de amortiguamiento a través de la siguiente ecuación dinámica:

$$\frac{dv_i}{dt} = -b_i \frac{v_i}{v_{0,i}}$$

VI. Curvas de Bézier

Una curva de Bézier es una curva polinomial que aproxima a una serie de puntos llamados puntos de control. Está definida por un conjunto de puntos de control P_0 a P_n donde n es su grado. Se dice que una curva de grado n aproxima a $n+1$ puntos de control. El primer y el último punto de control son siempre los puntos extremos de la curva; sin embargo, los puntos de control intermedios (si los hay) por lo general no se encuentran en la misma.

$$P(u) = \sum_{i=0}^{n} P_i B_i^n(u)$$
$$B_i^n(u) = \binom{n}{i}(1-u)^{n-i}u^i$$

donde P_i es el conjunto de puntos, $B_i^n(u)$ representa los polinomios de Bernstein y u toma valor entre 0 y 1.

Para producir una curva en nuestro mapa se crean las calles (rectas) y se utilizan las curvas de Bézier como un spline de suavizado. Nótese que esto solo se utiliza para la parte visual, en realidad los carros estarán corriendo en una sola calle.

IV. Inteligencia Artificial

I. Algoritmo genético

i.1. Optimización

Mediante el uso de un algoritmo genético se intenta minimizar las colas demoradas de

carros en las intersecciones.

i.2. Solución

La solución será un vector que contiene enteros correspondientes a los tiempos de luz verde de cada turno de la intersección, para todas las intersecciones del mapa.

i.3. Población inicial

La población inicial se hallará de manera aleatoria, en donde cada uno de los valores de tiempo de luz verde estará entre el tiempo promedio que le toma a un carro pasar por la intersección y el tiempo máximo que un vehículo puede estar esperando.

i.4. Función de evaluación *fitness*

Para evaluar qué tan buena es una solución, se ejecuta con ella una simulación. Luego, con los resultados arrojados por esta en un período determinado y bajo el criterio escogido, se le da una puntuación a dicho individuo.

Se consideraron tres formas de calcular el fitness:

- Según el máximo tiempo de espera.
- Según el tiempo total de paso de los carros por una calle.
- Según la media ponderada, calculada sobre todas las calles, del tiempo promedio que toma a los carros cruzar una calle.

Nótese que como se quieren minimizar los tiempos de los criterios anteriores, se tomarán dichos valores negativos para el fitness.

i.5. Selección de padres

Para escoger los padres se utilizó la selección basada en el rango (*rank selection*), en la cual se ordena según el valor de fitness y se da una probabilidad de selección a cada individuo.

La selección basada en el rango reduce los efectos potencialmente dominantes de individuos de alto fitness, comparativamente, en la población, estableciendo una cantidad predecible y limitada de presión de selección a favor de tales individuos. Al mismo tiempo, exagera la diferencia entre valores de fitness agrupados de forma cercana para que los mejores se puedan muestrear más.

i.6. Mutación

La función de mutación recibe una población y una probabilidad de mutación igual al inverso del número de individuos. Por cada gen de cada individuo, se escoge un número m aleatorio entre 0 y 1, y dicho gen es mutado si m es menor que la probabilidad de mutación. La forma de mutar es escoger un número aleatorio entre el tiempo promedio que le toma a un carro pasar por la intersección y el tiempo máximo que un vehículo puede estar esperando, de igual forma que para crear la población inicial.

i.7. Cruzamiento

La función de cruzamiento genera dos individuos para añadir a la nueva población a partir de los padres. Se consideraron tres formas de realizar el cruzamiento:

- **Cruzamiento por puntos múltiples:** Se seleccionan p puntos random y los segmentos (resultado de picar los individuos por dichos puntos) alternos de los individuos se intercambian para obtener nuevos descendientes.
- **Cruzamiento geométrico:** Se emparejan los genes que tienen la misma posición en los padres, y en esta posición del hijo se pone la media geométrica entre ellos. Además de este descendiente, se devuelve uno de los padres, escogido de forma aleatoria.
- **Cruzamiento intermedio:** Se emparejan los genes que tienen la misma posición en los padres, y en esta posición del hijo se pone la media aritmética entre ellos. Además de este descendiente, se devuelve uno de los padres, escogido de forma aleatoria.

II. A^*

La búsqueda A^* es un algoritmo de búsqueda *best-first* informada que utiliza la función de evaluación:

$$f(n) = g(n) + h(n)$$

donde $g(n)$ es el costo de alcanzar el nodo n, $h(n)$ es una heurística del costo estimado del camino de menor costo desde n hasta el objetivo, siendo $f(n)$ entonces el costo estimado de la mejor solución que pasa por n.

Se concibieron dos algoritmos A^*, el clásico para hallar la distancia mínima entre dos calles, y otro para hallar el camino que menor tiempo le tomará a un vehículo para llegar de una calle a otra.

ii.1. A^* para hallar la distancia mínima entre dos calles

$g(n) =$ *distancia recorrida hasta el momento.*

$h(n) =$ *distancia en línea recta desde el punto actual hasta el objetivo.*

ii.2. A^* para hallar el camino que menor tiempo le tomará a un vehículo para llegar de una calle a otra

$g(n) =$ *tiempo que demoró el vehículo en llegar desde el estado inicial hasta el estado intermedio n.*

Para hallar $g(n) =$ se corre una simulación con un vehículo que vaya desde el estado inicial hasta el estado intermedio n, y se toma el tiempo que demoró en llegar a su destino.

aclarar lo de que va soltando carros en cada esquina

$h(n) =$ *aproximado del tiempo requerido para llegar del estado intermedio n hasta el estado final.*

Se precacula, ejecutando Disjktra desde el destino hacia todos los puntos del mapa, el tiempo que tomará cada una de esas rutas, teniendo en cuenta los semáforos, distancia de las calles, velocidad de los carros y tráfico. Luego, para hallar $h(n)$ simplemente se toma el tiempo de la ruta del destino al nodo intermedio n.

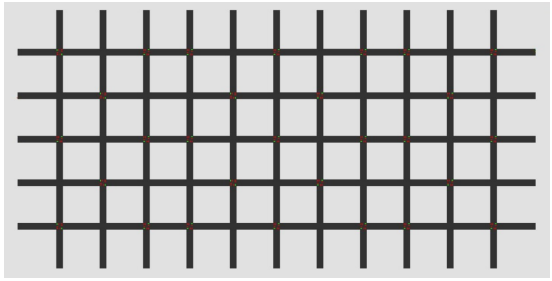
V. Tests

Se ejecutaron 50 pruebas con los siguientes parámetros para la optimización:

chequear cantidad de tests

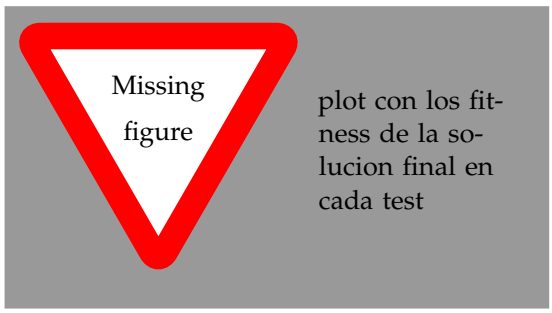
- **tamaño de la población:** 30
- **cantidad de iteraciones:** 50
- **tiempo de observación de cada individuo en la simulación:** 10s
- **velocidad de la simulación:** 30
- **tipo de cruzamiento:** cruzamiento por puntos múltiples (2)
- **cálculo de fitness:** según la media ponderada, calculada sobre todas las calles, del tiempo promedio que toma a los carros cruzar una calle
- **tiempo promedio que le toma a un carro pasar por la intersección:** 3s
- **tiempo máximo que un vehículo puede estar esperando:** 90s

sobre el mapa, de 12×6 cuadras y 29 intersecciones semaforizadas, que se muestra a continuación:



Dichas pruebas se pueden encontrar en la carpeta *tests* del repositorio.

I. Resultados



Explicar resultados

VI. Recomendaciones

Poner algun problema actual

Trabajo futuro: creacion de los mapas con gramatica prob. como habia pensado leo

Como trabajo futuro sería interesante optimizar la distribución de los turnos tal que en una intersección haya la menor cantidad posible de estos, y optimizar la posición de las intersecciones semaforizadas (añadir, quitar o reacomodar intersecciones en el mapa), para contribuir a hacer el tráfico más fluido, objetivo perseguido en este proyecto.

Todo list