Informe análisis de semáforos

Aaron Sarmiento
Robert Burgos
Nelson Rincon
Sergio Heredia
Laura García
Miguel Thómas

Materia:
Sistemas complejos

Universidad Sergio Arboleda

Escuela de Ciencias Exactas e Ingeniería
Ciencias de la Computación e Inteligencia Artificial
2024

Introducción

La gestión eficiente del tráfico automovilístico es un desafío constante debido al crecimiento automotor, que causa congestiones y largos tiempos de espera en cruces e intersecciones. Los sistemas tradicionales de semaforización, estáticos y predefinidos, a menudo resultan ineficaces ante cambios repentinos en el tráfico. Para abordar este problema, surge la idea de semáforos autoorganizados, que se basan en principios de autoorganización y adaptación emergente para regular el tráfico de manera más eficiente. Estos sistemas permiten una coordinación dinámica entre semáforos y una adaptación en tiempo real a las condiciones del tráfico, minimizando los tiempos de espera y optimizando el flujo vehicular.

Estructura del Código

El código implementado para la simulación de semáforos autoorganizados se organiza en tres clases principales que abarcan diferentes aspectos del sistema:

- **Clase Vehicle**

  Esta clase representa un vehículo en la simulación. Cada vehículo está definido por su posición, tamaño y dirección, y se encarga de gestionar su propia representación gráfica en la ventana de simulación, así como su movimiento dentro del entorno. La clase Vehicle facilita la visualización y el control de los vehículos durante la simulación.

```python
class Vehicle:
    def __init__(self, canvas, x, y, width, height, direction):
        self.canvas = canvas
        self.direction = direction
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.shape = None
        self.draw()

    def draw(self):
        if self.direction == "horizontal":
            self.shape = self.canvas.create_rectangle(self.x, self.y, self.x + self.width, self.y + self.height, fill="yellow")
        else:
            self.shape = self.canvas.create_rectangle(self.x, self.y, self.x + self.height, self.y + self.width, fill="blue")

    def move(self, x, y):
        self.canvas.move(self.shape, x, y)
        self.x += x
        self.y += y
```

- **Clase semáforo**

  La clase Semaforo representa un semáforo en la simulación. Cada semáforo tiene una posición, un radio y un color inicial. Esta clase permite cambiar dinámicamente el color del semáforo y verificar su estado (verde o rojo).

```python
class Semaforo:
    def __init__(self, canvas, x, y, radius, initial_color, direccion):
        self.vehiculos_cerca = 0
        self.contador=0
        self.direccion = direccion

        self.canvas = canvas
        self.x = x
        self.y = y
        self.radius = radius
        self.color = initial_color
        self.shape = self.canvas.create_oval(x - radius, y - radius, x + radius, y + radius, fill=initial_color)

    def change_colores(self, new_color):
        self.color = new_color
        self.canvas.itemconfig(self.shape, fill=new_color)

    def is_green(self):
        return self.color == "green"

    def is_red(self):
        return self.color == "red"
```

- **Clase Semaphore Simulation**

  La clase SemaphoreSimulation es la clase principal que controla la simulación. Se encarga de administrar la creación y movimiento de vehículos, así como de implementar la lógica de los semáforos auto organizados. Esta clase también gestiona hilos para permitir la ejecución simultánea de diferentes aspectos de la simulación, como el movimiento de los vehículos y la actualización de los semáforos. Además, en esta clase se inicializan todos los elementos gráficos de la simulación, como los semáforos, los vehículos y los cruces. Además, esta clase es responsable de gestionar la lógica de control de los semáforos, incluyendo la detección de vehículos cercanos y la adaptación de los tiempos de luz en función de las condiciones del tráfico.

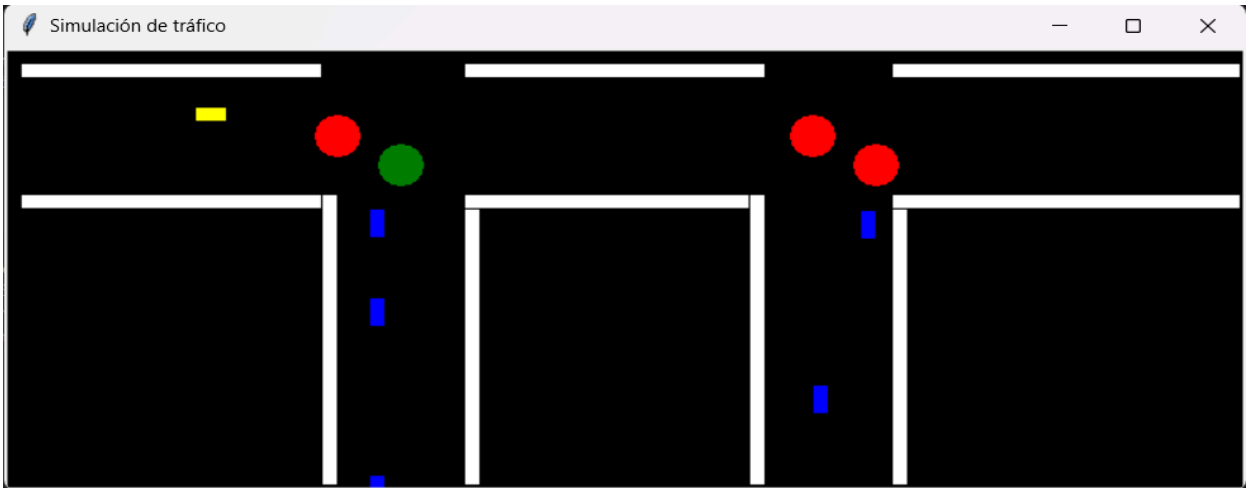
```python
class SemaphorSimulation:
    def __init__(self, master):···

    def on_canvas_click(self, event):···

    def run_semaphore(self):
        while True:
            self.contadorSemaforos =0

            # CADA QUE CAMBIA EL SEMAFORO DEBO CAMBIAR  semaforo.vehiculos_cerca = 0

            #CONDICIONES:

            # Recorre la lista de semáforos
            for semaforo in self.semaforos:
                # 1. En cada paso de tiempo, agregar a un contador el número de vehículos que
                # se acercan o esperan ante una luz roja a una distancia d. Cuando este contador exceda
                #un umbral_n, cambiar el semáforo (Siempre que el semáforo cambia reiniciar el contador a 0)
                if semaforo.is_red():
                    if(semaforo.direccion=="horizontal"):
                        # Recorre la lista de vehiculos horizontal
                        if(self.vehicles_horizontal!=None):
                            for vehicle in self.vehicles_horizontal:
                                if (semaforo.x > vehicle.x >= (semaforo.x -self.distancia_d)):
                                    semaforo.vehiculos_cerca +=1

                                if(semaforo.vehiculos_cerca > self.umbral_n):
                                    semaforo.vehiculos_cerca = 0
                                    #                 SEMAFORO, POSICIÓN SEMAFORO, HORIZONTAL, VERTICAL
                                    self.change_color(self.contadorSemaforos,"green","red")
                                    print("entra1")
                                    self.extra = True


                    else:
                        # Recorre la lista de vehiculos vertical
                        if(self.vehicles_vertical!=None):
                            for vehicle in self.vehicles_vertical:
                                if (semaforo.y < vehicle.y <=(semaforo.y + self.distancia_d)):
                                    semaforo.vehiculos_cerca +=1

                                if(semaforo.vehiculos_cerca > self.umbral_n):
```

Funcionamiento de la Semaforización Autoorganizada en el código

La semaforización autoorganizada es un enfoque dinámico y adaptable para controlar el tráfico en intersecciones, que se basa en la autoorganización de los semáforos en función de las condiciones del tráfico en tiempo real. La implementación realizada en el código refleja este concepto a través de una serie de condiciones y reglas que manejan el cambio de los semáforos. A continuación, se detalla cómo funciona la implementación en relación con las condiciones:

1. **Conteo de vehículos cercanos:**
    - Se realiza un seguimiento del número de vehículos que se acercan a una luz roja a una distancia específica. Esto se logra mediante la evaluación de la posición de los vehículos con respecto al semáforo en cuestión. Cuando el número de vehículos que se aproximan excede un umbral predefinido, se activa un cambio en el semáforo correspondiente, permitiendo que los vehículos atraviesen la intersección.
2. **Evitar cambios innecesarios:**
    - Para mantener un flujo de tráfico constante y evitar interrupciones innecesarias, se implementan condiciones que evitan cambios de semáforo cuando hay pocos vehículos cerca de una luz verde. Además, se considera la situación en la que un vehículo se detiene cerca de una luz verde. En este caso, el cambio de semáforo se pospone para evitar perturbar el flujo de tráfico.
3. **Detención por vehículos detenidos:**
    - Si un vehículo se detiene cerca de una luz verde, se activa un cambio en el semáforo para detener el flujo de tráfico en esa dirección y prevenir posibles colisiones.
4. **Cambio conjunto de luces:**
    - Cuando hay vehículos detenidos en ambas direcciones a una corta distancia más allá de la intersección, ambos semáforos se cambian a rojo simultáneamente. Esta medida se implementa para garantizar la seguridad en la intersección y evitar conflictos entre vehículos que se aproximan desde diferentes direcciones.
5. **Coordinación entre semáforos:**
    - Se coordina el cambio de los semáforos en direcciones opuestas para optimizar el flujo de tráfico. Si no hay vehículos que se acerquen a una luz verde en una dirección específica, pero al menos un vehículo se aproxima a una luz roja en la dirección opuesta, se activa un cambio en el semáforo para favorecer el movimiento de vehículos en esa dirección.

Ejecución de la simulación



1. **Interfaz Gráfica:**

   La simulación se presenta en una ventana gráfica creada utilizando la biblioteca Tkinter. En esta ventana, se representa una vista de la intersección donde ocurre la simulación del tráfico. Los elementos visuales incluyen los vehículos, los semáforos y los cruces de calles, los cuales son representados mediante formas geométricas como rectángulos y círculos.

2. **Generación y Movimiento de Vehículos:**

   Los vehículos se generan de manera aleatoria en la intersección y se mueven a lo largo de las calles de manera autónoma. Cada vehículo es una instancia de la clase Vehicle, que contiene información sobre su posición, tamaño y dirección de movimiento. Los vehículos interactúan con los semáforos y responden a los cambios de color, deteniéndose o avanzando según las señales luminosas.

3. **Lógica de los Semáforos:**

   La lógica de los semáforos se implementa en la clase SemaphoreSimulation. Dos hilos se utilizan para ejecutar simultáneamente la lógica de los semáforos y el movimiento de los vehículos. Los semáforos se representan como círculos de diferentes colores en la interfaz gráfica. Su estado (verde o rojo) se actualiza dinámicamente según las condiciones del tráfico y las reglas de semaforización autoorganizada.La clase Semaforo contiene métodos para cambiar el color del semáforo y verificar si está en verde o rojo.

4. **Hilos de Ejecución:**

   Se utilizan dos hilos para gestionar la simulación de manera concurrente:

   - Un hilo se encarga de ejecutar la lógica de los semáforos, evaluando las condiciones del tráfico y realizando los cambios de color según corresponda.

- El otro hilo se encarga de mover los vehículos por la intersección, actualizando continuamente sus posiciones en la ventana gráfica.

Conclusiones

- La solución propuesta utiliza un enfoque basado en reglas simples pero efectivas, como el conteo de vehículos cercanos, la coordinación entre semáforos y la detección de vehículos detenidos, para tomar decisiones de cambio de color de manera autónoma.

- La aplicación de la autoorganización en la regulación del tráfico a través de semáforos ha mostrado ser eficiente. La capacidad de los semáforos para adaptarse a las condiciones cambiantes del tráfico, evita cambios innecesarios y mejora la fluidez del tráfico.

- La implementación logra optimizar el flujo vehicular al cambiar dinámicamente los semáforos, considerando factores como la proximidad de vehículos y la presencia en ambas direcciones. Esto contribuye a una circulación más eficiente y segura.

- La semaforización autoorganizada contribuye a mejorar la seguridad vial al reducir el riesgo de colisiones y conflictos entre vehículos.