Nombre: Carlos Alberto Valladares Guerra **Carné:** 221164

Explicación de la Estrategia de Paralelización

GitHub: https://github.com/vgcarlol/MiniProyParalela

1. Breve descripción del problema

El proyecto consiste en simular el flujo de tráfico en una intersección con múltiples carriles y semáforos. En la simulación participan vehículos que avanzan por su carril dependiendo del estado del semáforo que les corresponde. El estado de cada semáforo cambia cíclicamente entre ROJO, AMARILLO y VERDE, con duraciones predefinidas para cada fase. La simulación se ejecuta durante un número fijo de iteraciones, mostrando en cada paso la posición de los vehículos y el estado de los semáforos.

2. Estrategia de paralelización utilizada

Para optimizar el rendimiento, se implementó una versión paralela con OpenMP. Se identificaron dos tareas principales que pueden ejecutarse de forma concurrente:
- Actualización de semáforos.
- Movimiento de vehículos.

Estas tareas son independientes y se paralelizaron mediante la directiva `#pragma omp parallel sections`, permitiendo que cada una se ejecute en su propio conjunto de hilos.

Además, dentro de cada tarea se aplicaron otras técnicas de paralelización:

- En la actualización de semáforos, se usó `#pragma omp parallel for schedule(static)` para repartir uniformemente el trabajo, ya que el costo por semáforo es similar.
- Se incluyó un ejemplo de paralelismo anidado para simular subtareas dentro de la gestión de cada semáforo.
- En el movimiento de vehículos, se usó `#pragma omp parallel for schedule(dynamic, 16)` debido a que el trabajo por vehículo puede variar según su posición y el estado del semáforo.
- Para evitar condiciones de carrera al mover varios vehículos hacia la misma posición, se implementó un arreglo de ocupación y operaciones atómicas (`#pragma omp atomic`).

3. Justificación del uso de OpenMP

OpenMP fue elegido por su facilidad de integración en programas escritos en C y por permitir añadir paralelismo de forma incremental mediante directivas.

Se emplearon varias características clave de OpenMP:

- **Paralelismo dinámico:** (`omp_set_dynamic(1)`): permite que el número de hilos activos se ajuste automáticamente según la carga de trabajo, optimizando el uso de recursos.
- **Paralelismo anidado:** (`omp_set_nested(1)`): habilita la creación de regiones paralelas dentro de otras, útil para dividir tareas complejas en sub-tareas paralelas.
- **Parallel sections:** ideal para ejecutar en paralelo tareas independientes como la actualización de semáforos y el movimiento de vehículos.
- **Parallel for** con políticas de planificación (`schedule(static)` y `schedule(dynamic)`): proporciona control sobre cómo se reparte el trabajo entre hilos, adaptándose al tipo de tarea.