Informe de progreso del TFG

20 de marzo de 2020

Elaborado por Francisco A. Cruz Zelante
Para Eduardo Segredo, tutor; y Gara Miranda, cotutora

1. Resumen

- En principio, y salvo pequeños cambios, la red de la rotonda de Anchieta está preparada. Se han modificado las intersecciones para ajustarlas adecuadamente junto con los pasos de peatones y se han modificado manualmente las fases de los semáforos en algunas intersecciones para mejorar la circulación. Por ahora el flujo de vehículos se ha creado aleatoriamente; en un futuro debe hacerse respecto a los datos provistos por el Cabildo.
- Con respecto al código. Ahora mismo soy capaz de leer y procesar el XML de la red, extraer datos (id, duración y fases) de los semáforos, convertir esos datos (fenotipo) al genotipo que necesita el algoritmo evolutivo, que este se ejecute, y luego realizar el paso inverso para que, una vez que el algoritmo nos devuelva el genotipo ya modificado, procesar este a un fichero XML legible por SUMO para que realice la simulación y calcular el fitness del individuo. Básicamente, la *pipeline* del algoritmo evolutivo está hecha, pero necesita mejoras.
- **BUG.** Parece ser que la librería Genetics.js, programada por Cristian Abrante, tiene un bug que reinicia el rango en el cual se generan los valores de las duraciones en el algoritmo evolutivo después de la primera generación. Más información en el apartado correspondiente.

2. Sobre el archivo de red y las fases de los semáforos

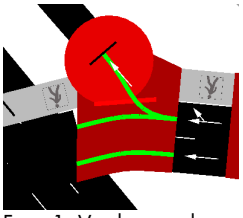
Ajustes en las intersecciones y las fases

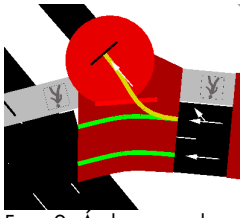
Por ahora, creo que tengo un mapa de Anchieta relativamente bueno. Pasé varias horas intentando ajustar correctamente las entradas y salidas de la rotonda para que los semáforos tuvieran sentido con los pasos de peatones. Porque como hablamos la última vez, aunque los peatones no tengan por qué simularse en SUMO, los *pasos* de peatones llevan aparejada una luz semafórica que tiene que ir sincronizada con el semáforo de los coches que pasan por ese paso de peatones. De lo contrario, si un coche tiene luz verde pero el paso de peatones no está regulado por semáforos podría darse un accidente.

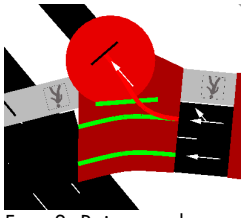
Hechas las intersecciones, empecé a añadir semáforos. SUMO no hace un mal

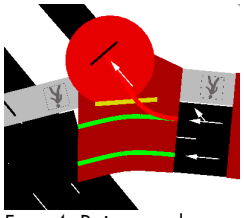
trabajo asignando las fases automáticamente, pero tuve que modificar varias de ellas manualmente del siguiente modo:

Considera la salida que da a la incorporación de la TF5 dirección norte. SUMO, por defecto, hace que esté verde el semáforo peatonal y en rojo los dos carriles de la rotonda, y viceversa, cosa que realmente es ineficiente. Lo modifiqué manualmente para que, en esa salida, los dos carriles de la rotonda estén en verde siempre, y solo se pongan en rojo la salida en sí.


Fase 1: Verde para la salida, rojo para peatón.


Fase 2. Ámbar para la salida, rojo para el peatón.


Fase 3. Rojo para la salida, verde para el peatón.


Fase 4. Rojo para la salida, ámbar para el peatón (es decir, verde parpadeante).

Varias otras intersecciones se han modificado en la misma línea, teniendo en cuenta siempre los factores que puedan afectar a cómo se configuran las fases (pasos de peatones, cantidad de carriles, diseño de la propia intersección, etc.). En mi TFG final hablaré con detalle sobre todas las modificaciones realizadas a cada intersección.

Al final, lo que he procurado es que el carril interior de la rotonda siempre estuviera en verde (salvo para los casos de las entradas con dos carriles, puesto que ahí tiene que parar por el que viene de la entrada en el carril izquierdo).

Duración del ámbar

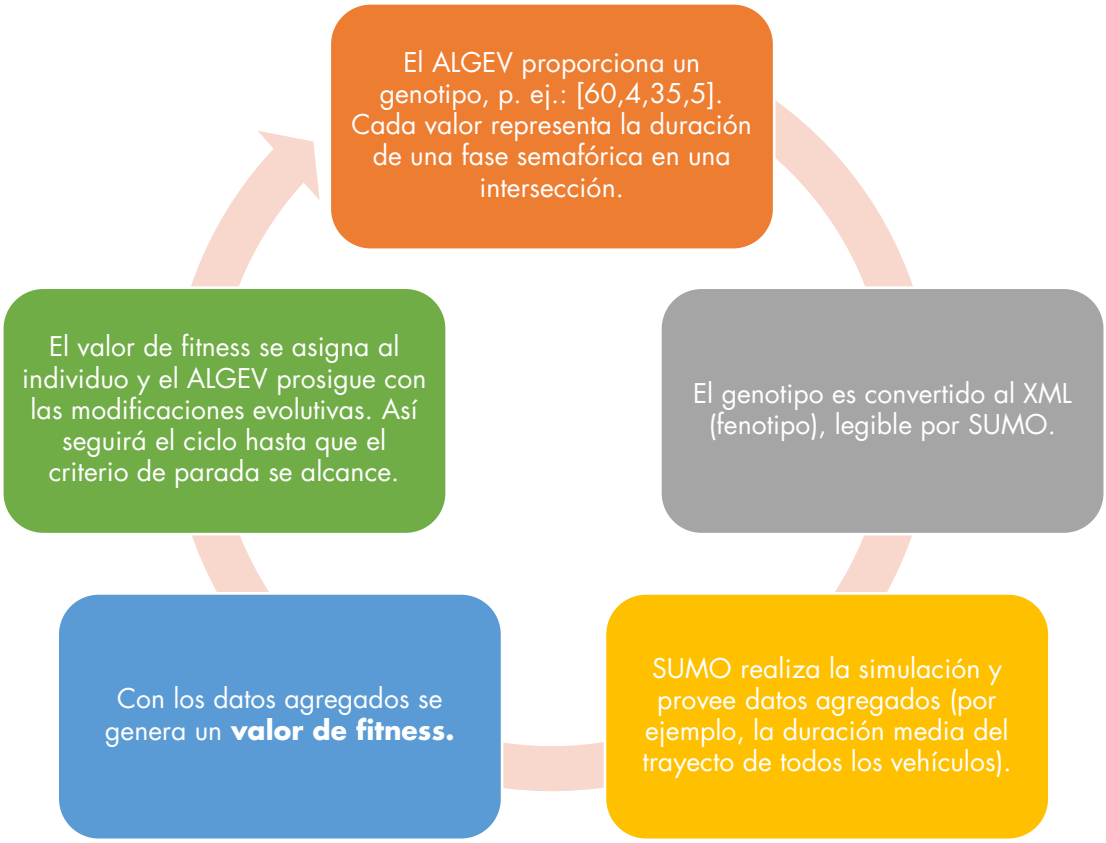
Otro detalle sobre el que también es necesario buscar información es sobre la duración que debe tener la fase ámbar de los semáforos. Por internet he conseguido varios artículos que tengo pendientes de leer [1], entre ellos algunos del *Institute of Transportation Engineers* (ITE), que parece ser una autoridad sobre seguridad vial y que tiene algunos artículos interesantes al respecto.

También he leído que la DGT ha emitido algunas recomendaciones sobre la duración que debería tener cada fase de los semáforos [2], pero en su web, por desgracia, no he conseguido nada. Quizás esté en alguna revista de la DGT que no está indexada.

Por ahora en varias noticias he leído que lo recomendado es establecer una duración de las fases (incluida la del ámbar) entre 35 y 150 segundos. No he conseguido la fuente primaria de tal afirmación.

3. Sobre el código

Cómo dije en el resumen, la base del algoritmo está hecha. Me refiero a lo siguiente (ALGEV es un acrónimo para *algoritmo evolutivo):*



A este respecto, los archivos fuente son los siguientes:

- `tl-logic.ts`: Contiene la clase `TLLogic` que representa una intersección con semáforos, sus fases y las duraciones de estas.
- `converter.ts`: Permite convertir un genotipo (array de números) a la clase `TLLogic`, y viceversa.
- `executor.ts`: Se encarga de ejecutar SUMO con los parámetros provistos. También contiene una interfaz para representar la salida que emite SUMO con los datos agregados de la simulación.
- `xml-io.ts`: Permite leer un fichero de tipo de red del tipo ".net.xml" y convertir los datos de los semáforos a un array de `TLLogic`. Y viceversa.
- `tl-logic.ts`: Punto de entrada del código. Contiene lo necesario para ejecutar el algoritmo evolutivo.

Otro detalle es que también estoy procurando que todas las librerías que empleo tengan una licencia de código abierto, del tipo MIT o GPL.

Bug en la librería Genetics.js

Me he encontrado con el siguiente bug en la librería Genetics.js que no sé cómo solucionar.

Por las características del problema del TLSP, el genotipo es un array de números del tipo [60, 4, 35, 5]. Para poder realizar las mutaciones, es necesario introducir aleatoriedad, pero siempre dentro de un rango. No tiene sentido introducir una duración de 234876276 segundos, por ejemplo; ni tampoco duraciones negativas. Precisamente por esto la librería contiene una clase, `NumericRange`, que me permite introducir un rango en el cual quiero que se generen los nuevos valores.

La cuestión es que este rango funciona según lo esperable, generando valores entre 4 y 90, pero solo en la primera generación. En la segunda generación se reinicia a los valores (-inf, inf), lo que me genera genotipos como [60, 5, -6908527863, 5]. Obviamente, SUMO falla no solo porque sea una duración excesiva, sino porque también es negativa.

Intentaré revisar mi código (por si el fallo fuera mío) y el código de la librería para ver si encuentro la causa que hace que se reinicie el rango, pero teniendo en cuenta que el código de la librería es de tamaño medio, y con varias abstracciones de por medio, podría llevarme bastante tiempo encontrar una solución. Hablaré con Cristian a ver si es posible que me ayude con esto.

En otro caso, las alternativas son:

1. Buscar otra librería de algoritmos evolutivos.
2. Elaborar mi propio algoritmo evolutivo, *ad hoc*. Algo que estoy considerando por el punto que expondré a continuación.

Rendimiento

Actualmente, el cuello de botella se encuentra en la duración de la simulación en SUMO. Con algo menos de 600 vehículos (algo que creo que tendría que incrementarse), tarda 10 seg. en ejecutarse. Esto se puede agilizar realizando la simulación de los individuos en paralelo, algo que mi tutor y yo habíamos comentado en nuestra última reunión (también con la utilización de semillas distintas, para variar el comportamiento aleatorio).

Sin embargo, tendría que revisar la librería Genetics.js para implementarlo, al ser esta la que se encarga de realizar las llamadas al ejecutor de SUMO. Lo puedo intentar, aunque quizás tener en cuenta un algoritmo evolutivo *ad hoc* puede ser una mejor idea.

4. Conclusiones y trabajo futuro

1. Revisar el archivo de red por si hubiera mejoras aplicables a la configuración de las fases de los semáforos. Comentar esto con los tutores.
2. Buscar fuentes primarias que hablen de la duración recomendada del ámbar.
3. Ajustar el flujo de vehículos según los datos provistos por el Cabildo.
4. Mejorar el código actual para que:
   a) sea un poco más robusto (control de errores),
   b) sea más legible, y

c) esté mejor documentado.

5. Explorar opciones con el bug de Genetics.js:
   a) Intentar arreglarlo.
   b) Buscar otra librería.
   c) Elaborar mi propio algoritmo evolutivo.
6. Sobre el cuello de botella en la ejecución de SUMO, explorar la posibilidad de:
   a) Implementar paralelismo en la librería Genetics.js.
   b) Elaborar mi propio algoritmo evolutivo.
7. Empezar a redactar la memoria documentando todo lo hecho hasta ahora.

5. Referencias

[1] J. Pastor, «Este ingeniero luchó contra la luz ámbar del semáforo y ganó: ahora organismos internacionales plantean ampliar su duración», *Xataka*, 26-oct-2019. [Online]. Disponible en: https://www.xataka.com/vehiculos/este-ingeniero-lucho-luz-ambar-semaforo-gano-ahora-estados-unidos-plantean-alargar-esa-duracion. [Accedido: 18-mar-2020]

[2] «La duración del ámbar en Madrid se aleja de lo recomendado por la DGT ⋆ AEA», *AEA*, 27-feb-2013. [Online]. Disponible en: https://aeaclub.org/duracion-ambar-semaforos-madrid/. [Accedido: 20-mar-2020]